ELSEVIER

Artificial Intelligence 113 (1999) 203–245

Artificial Intelligence

www.elsevier.com/locate/artint

LAZY propagation: A junction tree inference algorithm based on lazy evaluation

Anders L. Madsen *, Finn V. Jensen [1]

Department of Computer Science, Aalborg University, Fredrik Bajers Vej 7C, DK-9220 Aalborg, Denmark

Received 11 August 1999

Abstract

In this paper we present a junction tree based inference architecture exploiting the structure of the original Bayesian network and independence relations induced by evidence to improve the efficiency of inference. The efficiency improvements are obtained by maintaining a multiplicative decomposition of clique and separator potentials. Maintaining a multiplicative decomposition of clique and separator potentials offers a tradeoff between off-line constructed junction trees and on-line exploitation of barren variables and independence relations induced by evidence.

We consider the impact of the proposed architecture on a number of commonly performed Bayesian network tasks. The tasks we consider include cautious propagation of evidence, determining a most probable configuration, and fast retraction of evidence a long with a number of other tasks. The general impression is that the proposed architecture increases the computational efficiency of performing these tasks.

The efficiency improvement offered by the proposed architecture is emphasized through empirical evaluations involving large real-world Bayesian networks. We compare the time and space performance of the proposed architecture with non-optimized implementations of the HUGIN and Shafer–Shenoy inference architectures.

Keywords: Bayesian networks; Junction trees; Probabilistic inference

1. Introduction

The framework of Bayesian networks is an increasingly popular knowledge representation framework for reasoning under uncertainty. The most common task performed on a Bayesian network is calculation of the posterior marginal distribution for all non-evidence

* Corresponding author. Email: anders@cs.auc.dk.

[1] Email: fvj@cs.auc.dk.

0004-3702/99/$ – see front matter
PII: S0004-3702(99)00062-4

variables given a set of evidence. The complexity of inference in Bayesian networks is, however, known to be NP-hard [2]. The best known architectures for computing all posterior marginals in a Bayesian network are the Lauritzen–Spiegelhalter [18], the Shafer–Shenoy [27], and the HUGIN [15] architectures and various variations over these architectures (see, e.g., [26] and [13]). It has for a long time been a puzzle why these standard inference architectures for Bayesian networks based on secondary computational structures do not really use the direction of the edges in the original network. The standard architectures are based on a secondary structure (a junction tree or a join tree) build by triangulating the (moralized) graph of a Bayesian network. This secondary tree structure can be used for propagation for all information scenarios. Therefore, the architectures do not exploit independence relations induced by the evidence. That is, the tree structure is large enough to take care of all instantiations of variables. For some (or sometimes all) specific information scenarios, a careful exploitation of the d-separation properties and barren variables would result in less complex structures.

Consider for example the Bayesian network indicated in Fig. 1. If X is instantiated and no evidence has been entered to DAG_4, then DAG_1, DAG_2, and DAG_3 are independent, and we need only pass messages down to DAG_4. An on-line triangulation of this scenario will result in a set of much simpler junction trees than an off-line produced junction tree. To exploit the specific independence relations induced by the evidence on X, we need a very efficient algorithm for detecting independence relations and performing an efficient triangulation based on these independence relations. In particular, as the problem of optimal triangulation is NP-hard, there is not much hope that an architecture requiring on-line triangulation can outperform the above referred architectures for large networks, and improved performance for small networks is not particularly interesting. The updating task may be relaxed such that only updated probabilities for a very small set of variables given a set of evidence are of interest. For example, we might be interested in computing a single marginal given a set of evidence. In that case inference algorithms based on exploiting the structure of the graph of the Bayesian network directly often will be more time and space efficient than junction tree propagation algorithms. The standard inference algorithms for computing single marginals are the SPI [19], variable elimination [32], and bucket elimination algorithms [8]. These direct computation algorithms utilize barren variables and specific independence relations induced by evidence in the Bayesian network

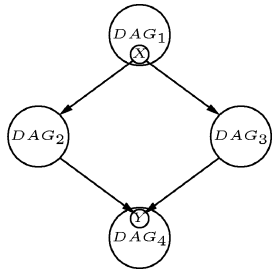

Fig. 1. If X is instantiated and no evidence has been entered to DAG_4, then it is only necessary to pass messages down to DAG_4.

to simplify the updating task. The direct computation algorithms are, however, not able to compute posterior marginals for all variables efficiently.

In this paper we propose a compromise between off-line triangulation and on-line exploitation of specific independence relations. We call the architecture LAZY *propagation* as the bulk of the algorithm is lazy evaluation of the messages passed between the cliques and separators of the junction tree. The inference algorithm is based on finding a partial variable elimination order off-line and then extending this partial order to a total order on-line. The partial elimination order is determined by constructing a junction tree representation of the original Bayesian network. The conditional probability distributions of the Bayesian network are associated with cliques of the junction tree, but instead of combining the distributions associated with the cliques, it is on-line determined which potentials to combine when a message is to be computed. Thereby, when a message is to be computed only the required potentials are combined. An effect of this scheme is that barren variables and independence relations induced by evidence are exploited.

2. Preliminaries and notation

In this section we are for ease of reference going to briefly review results published by others and to specify the notation used throughout the paper. We expect the reader to be familiar with most of the results reviewed in this section.

A Bayesian network $N = (G, \mathcal{P})$ consists of a directed acyclic graph $G = (V, E)$ and a multiplicative decomposition $\mathcal{P}$ of the joint probability distribution over the variables of N. The variables of N are all assumed to be discrete. The nodes V of the graph G corresponds one-to-one with the variables of N. Hence, we will refer to nodes and variables interchangeably. The joint probability distribution $P(V)$ is assumed to decompose multiplicatively according to the graph G (assumed to be connected) such that:

$$P = \prod_{X \in V} P(X|pa(X)),$$

where $pa(X)$ is the set of parent variables of X.

The usual inference task performed when considering Bayesian networks is repetitive computation of all single posterior marginals given different sets of evidence. The architectures considered to be most efficient for solving this task are all based on construction of a secondary computation structure from the Bayesian network.

If, on the other hand, the updating task is relaxed to the computation of the posterior probability distribution of a few non-evidence variables only, then direct computation algorithms tend to be most efficient. One of the properties of Bayesian networks utilized by direct computation algorithms is referred to as *barren variables*. The concept of barren variables was introduced by Shachter [25]:

Definition 2.1. A variable is said to be a *barren variable* if it is neither an evidence nor a target variable and it only has barren descendants.

According to the above definition no variables are barren in junction tree based architectures as the underlying task is the calculation of the posterior marginal probability

distribution for all non-evidence variables in the Bayesian network. The property of barren variables exploited by direct computation algorithms is that barren variables have no impact on the posterior probability distribution of the target set. The concept of barren variables is a consequence of one of the basic axioms of probability theory:

Axiom 2.2 (*Unity-potential axiom*).

$$\sum_H P(H|T) = 1_T.$$

Notice that the unity-potential axiom also applies, if H is a set of variables.

2.1. Evidence

In this paper we differentiate between two kinds of evidence. *Hard evidence* is instantiation of a variable and *soft evidence* is a finding or a likelihood potential on a variable. A *finding potential* ε_X on a variable X is a potential of X with values zero and one. The finding potential ε_X indicates the possible and impossible states of X:

$$\varepsilon_X(x) = \begin{cases} 1 & \text{if } x \text{ is a possible state,} \\ 0 & \text{otherwise.} \end{cases}$$

For each impossible state of X, the finding potential takes the value zero and for each possible state of X, the finding potential takes the value one. As opposed to a finding potential a *likelihood potential* weights the possible states of the evidence variable. In this paper we will not consider soft evidence further, except if its explicitly stated. This implies that whenever we refer to evidence we will implicitly assume this to be hard evidence.

The independence relations induced by a set of evidence in a Bayesian network can be determined using the d-separation criteria:

Definition 2.3 (*d-separation*). Variables X and Z in a Bayesian network $N = (G, \mathcal{P})$ are d-separated if for all paths connecting X and Z there is a intermediate variable Y such that one of the following statements is satisfied:

- Y is the middle variable in a serial or a diverging connection, and Y is instantiated by evidence.
- Y is the middle variable in a converging connection, and neither Y nor any of its descendants have received evidence.

We say that two variables in a Bayesian network are d-connected, if they are not d-separated.

Geiger et al. [9] present a graph based algorithm for determining the set of variables R which are d-separated from another set of variables J given a set of variables L. It is straightforward to adjust the algorithm such that the set of variables R returned is the set of variables d-connected to J given L.

The algorithm for determining the set of relevant variables is based on another simple algorithm for determining the set of variables reachable from a set of variables J given a set of illegal pairs of directed edges:

Algorithm 2.1 (*Find reachable variables*). Let $G = (V, E)$ be a directed graph, let F be a set of illegal pairs of directed edges, and let J be a set of variables. To determine the set of variables R d-connected to J do:

(1) Set $R = \emptyset$.
(2) Add a new variable S to V.
(3) **For** each $X \in J$
  (a) Add a directed edge from S to X and label it with 1.
  (b) Add X to R.
(4) Label all other edges with "undefined".
(5) Set $i = 1$.
(6) **Repeat**
  (a) **For** each unlabeled directed edge (Y, Z) adjacent to at least one directed edge (X, Y) labeled i such that $((X, Y), (Y, Z))$ is a legal pair
    (i) Label (Y, Z) with $i + 1$.
    (ii) Add Z to sR.
  (b) Set $i = i + 1$.
  Until no legal pair of directed edges $((X, Y), (Y, Z))$ exists.
(7) Return R.

Algorithm 2.1 is used to determine the set of nodes R d-connected to a set of nodes J given L:

Algorithm 2.2 (*Find the set of d-connected nodes*). Let $G = (V, E)$ be a directed acyclic graph and let J and sL be sets of nodes. To determine the set of nodes $R = \{X \mid X$ d-connected to $Y \in J$ given $L\}$ do:

(1) Construct the table:

$$\mathit{descendent}[X] = \begin{cases} \mathit{true} & \text{if } X \text{ is or has a descendent in } L, \\ \mathit{false} & \text{otherwise.} \end{cases}$$

(2) Construct a directed graph $G' = (V, E')$ where $E' = E \cup \{(X, Y) \mid (Y, X) \in E\}$.
(3) Let $F^{\complement}$ be the set of legal pairs of directed edges where a pair of edges $((X, Y), (Y, Z))$ is legal if and only if $X \neq Z$ and either of the following holds:
  (a) The node Y is not a head-to-head node on the path $X - Y - Z$ in G and $Y \notin L$, or
  (b) The node Y is a head-to-head node on the path $X - Y - Z$ in G and $\mathit{descendent}[Y]$.
(4) Invoke Algorithm 2.1 to determine the set of nodes R reachable from J by a legal path in G'.
(5) Return R.

The algorithm has time complexity linear in the number of edges of the graph as each edge is visited a constant number of times.

2.2. Direct computation

A *query* $Q = (T, \varepsilon)$ on a Bayesian network consists of a set of variables T and a set of evidence ε. The set T is referred to as the *target* set as the answer to the query is the posterior joint probability distribution of T given ε.

Direct computation algorithms for answering a specific query $Q = (T, \varepsilon)$ on a Bayesian network $N = (G, \mathcal{P})$ use the structure of the graph G to simplify the task of computing the answer as much as possible. The task is simplified by pruning from G all nodes d-separated from T given ε and all nodes corresponding to barren variables. Let $G' = (V', E')$ be the pruned graph and let $N' = (G', \mathcal{P}')$ be the Bayesian network corresponding to G' where $\mathcal{P}' \subseteq \mathcal{P}$ consists of the conditional probability distributions corresponding to sV'. The answer to Q is obtained by eliminating from N' all variables not included in the query. The joint potential of T and ε is computed as:

$$\phi(T, \varepsilon) = \sum_{X \in sV' \setminus T} \prod_{Y \in V'} P(Y|pa(Y)). \tag{1}$$

The conditional probability distribution $P(T|\varepsilon)$ can be obtained by normalization.

There exists at least two different direct computation approaches to computing the potential $\phi(T, \varepsilon)$ of Eq. (1). The calculations performed to obtain $\phi(T, \varepsilon)$ can be arranged in a computation tree where the root corresponds to $\phi(T, \varepsilon)$, each leaf corresponds to a potential of $\{P(Y|pa(Y)) \mid Y \in V'\}$, and an internal node corresponds to either a combination of potentials or a marginalization of a variable.

One approach is to let the construction of the computation tree be driven by variable elimination. That is, the task of computing $\phi(T, \varepsilon)$ is considered as a problem of eliminating variables where the potentials including the variable X have to be combined before X can be eliminated. Direct computation algorithms such as bucket elimination [8], variable elimination [31], the peeling method [1], the fusion operation [28] use the approach based on variable elimination.

The other approach is to let the computation tree construction be driven by combination of potentials. That is, the task of computing $\phi(T, \varepsilon)$ is considered as a problem of combining potentials where a non-target variable X is eliminated when X is only contained in a single potential. The SPI algorithm [19] uses the approach based on combination of potentials.

Notice that none of the two approaches is always better than the other.

2.3. Junction trees

A junction tree representation of a Bayesian network $N = (G, \mathcal{P})$ is constructed by moralization and triangulation of G. The nodes of the junction tree correspond to

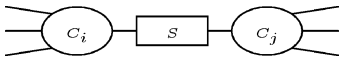

Fig. 2. Two adjacent cliques C_i and C_j separated by the neighbouring separator S.

cliques—maximal complete subgraphs—of the triangulated graph. The cliques of the junction tree are connected by separators such that the so-called junction tree property holds. The junction tree property insures that whenever two cliques C_i and C_j are connected by a path, the intersection $S = C_i \cap C_j$ is a subset of every clique and separator on the path (see Fig. 2). A junction tree also has the property that each variable and its parents are contained in at least one clique. The set of neighbouring separators of a clique C is referred to as $ne(C)$ while two cliques connected by a separator are said to be adjacent.

Every conditional probability distribution of the original Bayesian network (i.e., $P(X|pa(X)),\ \forall X \in V$) is associated with a clique such that the domain of the distribution is a subset of the clique domain (we use the notation $dom(\phi)$ to refer to the domain of a potential ϕ). The set of distributions Φ_C associated with a clique C are in standard junction tree architectures combined to form the initial clique potential ϕ_C:

$$\phi_C = \prod_{\phi \in \Phi_C} \phi.$$

A separator S is said to be *consistent*, if it contains the information its two neighbouring cliques C_i and C_j have in common. That is, a separator is consistent if:

$$\sum_{C_i \setminus S} \phi_{C_i} = \phi_S = \sum_{C_j \setminus S} \phi_{C_j},$$

where ϕ_S is the separator potential, ϕ_{C_i} is the clique potential for C_i, and ϕ_{C_j} is the clique potential for C_j. If all separators of a junction tree are consistent, then the junction tree is said to be consistent.

There are two aspects to handling evidence in a junction tree. First, the evidence needs to be inserted into the junction tree and second the evidence needs to be propagated throughout the junction tree to obtain consistency. In standard junction tree inference architectures evidence on a variable is associated with a single clique containing the evidence variable. Evidence can be incorporated by either associating a finding potential with the appropriate clique or changing the entries of the clique potential to reflect the evidence. If we in the later case, for example, have evidence $X = x$, then all entries in the clique potential corresponding to $X \neq x$ are changed to zero and the entries corresponding to $X = x$ are left unchanged.

2.4. Message passing

Inference in junction tree based architectures is performed by passing messages between adjacent cliques. There exists at least two different approaches to message passing in junction trees. One approach is to choose a predetermined clique R as root of the junction tree and then pass messages in two different phases relative to R. During the first phase evidence is recursively collected to R and during the second phase evidence is

recursively distributed from R. Another approach to message passing is to pass messages asynchronously between cliques. A clique C_j is allowed to pass a message to an adjacent clique C_i, if C_j has received messages from all adjacent cliques except possibly C_i. Asynchronous message passing terminates when one message has been passed in each direction over all separators of the junction tree. It is straightforward to show that the same set of messages are passed regardless of the message passing approach taken. Messages can be passed independently in different branches of the junction tree.

Both approaches induce an orientation on the structure of the junction tree. In the asynchronous message passing scheme the root clique is the first clique to receive messages from all adjacent cliques while the root clique is trivially determined in the other approach. In both approaches the first cliques to pass messages are the leaf cliques.

While the message passing scheme for standard junction tree algorithms are similar, the calculation of the individual messages is different. We will give brief descriptions of how messages are calculated in the HUGIN and Shafer–Shenoy architectures. We will use junction trees as the computational structure for Shafer–Shenoy propagation even though this might not be the optimal computational structure for Shafer–Shenoy propagation.

2.4.1. HUGIN *messages*

In the HUGIN architecture each separator holds a single potential over the separator variables which initially is a unity potential. During propagation of evidence the separator and clique potentials are updated. Consider once more the two adjacent cliques C_i and C_j as shown in Fig. 2. When a message is passed from C_j to C_i either during collection or distribution of evidence, C_i absorbs evidence from C_j. Absorption of evidence in the HUGIN architecture involves performing the following calculations:

(1) Calculate the updated separator potential: $\phi_S^* = \sum_{C_j \setminus S} \phi_{C_j}$.

(2) Update the clique potential of C_i: $\phi_{C_i}^* = \phi_{C_i} \frac{\phi_S^*}{\phi_S}$.

(3) Associate the updated potential with the separator: $\phi_S = \phi_S^*$.

After a full round of message passing the potential associated with any clique (separator) is the joint probability distribution (up to the same normalization constant) of the variables in the clique (separator).

2.4.2. Shafer–Shenoy messages

In the Shafer–Shenoy architecture each separator is initially empty. During inference each separator is updated to hold each of the potentials passed over the separator. The clique potentials are, on the other hand, left unchanged. When evidence is absorbed from C_j to C_i, the potential ϕ_S^* passed over the separator S connecting C_i and C_j is calculated as:

$$\phi_S^* = \sum_{C_j \setminus S} \phi_{C_j} \prod_{S' \in ne(C_j) \setminus \{S\}} \phi_{S'}.$$

After a full round of message passing, the joint probability distribution (up to the same normalization constant) of any clique C_i in the junction tree can be computed as

the combination of the clique potential and all the received potentials associated with neighbouring separators:

$$\phi^*_{C_i} = \phi_{C_i} \prod_{S \in ne(C_i)} \phi_S.$$

For each separator the joint probability distribution (up to the same normalization constant) of the separator variables can be obtained by combining the two messages passed over the separator.

2.5. Posterior marginals

From a consistent junction tree the posterior marginal distribution of a variable and the evidence is readily computed. The posterior marginal distribution of a variable X and the evidence ε can be computed from any clique or separator potential ϕ containing X by eliminating all variables in $dom(\phi)$ except X [16]:

$$P(X, \varepsilon) = \sum_{Y \in dom(\phi) \setminus \{X\}} \phi.$$

The marginal distribution of X given ε is computed by normalization (notice that $P(\varepsilon = \sum_X P(X, \varepsilon))$.

2.6. The marginalization algorithm

The algorithm for eliminating a variable X from a set of potentials Φ by marginalization we will use is:

Algorithm 2.3 (*Marginalization*). Let $\Phi = \{\phi_1, \ldots, \phi_n\}$ be a set of potentials. If *marginalization* of X is invoked on Φ, then:

(1) Set $\Phi_X = \{\phi \in \Phi \mid X \in dom(\phi)\}$.
(2) Calculate $\phi^*_X = \sum_X \prod_{\phi \in \Phi_X} \phi$.
(3) Let $\Phi^* = \{\phi^*_X\} \cup \Phi \setminus \Phi_X$.

Φ^* is the resulting set of potentials when X is eliminated from Φ.

3. LAZY propagation

The LAZY propagation architecture is based on message passing. Any computational tree structure maintaining the (in)dependence relations of the Bayesian network can be used as the underlying computational structure of LAZY propagation. For ease of exposition we present the LAZY propagation architecture as a junction tree based inference architecture. We will assume that the message passing is controlled by selecting a predetermined clique as root of the junction tree. The bulk of LAZY propagation is to maintain a multiplicative decomposition of clique and separator potentials and to postpone combination of potentials. This gives opportunities for exploiting barren variables and independence relations induced by evidence during inference.

A detailed, formal description of the tasks involved when performing inference in the LAZY propagation architecture is presented in the following sections.

3.1. Message passing

Collection of evidence is performed recursively by invoking the COLLECTEVIDENCE algorithm on the predetermined root clique of the junction tree. When COLLECTEVIDENCE is invoked on a clique C_j from an adjacent clique C_i, then C_j invokes COLLECTEVIDENCE on all other adjacent cliques. When these cliques have finished their COLLECTEVIDENCE, C_i absorbs the message from C_j:

Algorithm 3.1 (COLLECTEVIDENCE). Let C_i and C_j be adjacent cliques. If COLLECTEVIDENCE is invoked on C_j from C_i, then:
(1) C_j invokes COLLECTEVIDENCE on all adjacent cliques except C_i.
(2) The message from C_j to C_i is absorbed by C_i (Algorithm 3.3).

At the end of the COLLECTEVIDENCE phase evidence is distributed from the predetermined root clique by recursively invoking the DISTRIBUTEEVIDENCE algorithm. When DISTRIBUTEEVIDENCE is invoked on a clique C_j from a neighbour C_i, C_j absorbs evidence from C_i and invokes DISTRIBUTEEVIDENCE on all other adjacent cliques.

Algorithm 3.2 (DISTRIBUTEEVIDENCE). Let C_i and C_j be adjacent cliques. If DISTRIBUTEEVIDENCE is invoked on C_j from C_i, then:
(1) The message from C_i to C_j is absorbed by C_j (Algorithm 3.3).
(2) C_j invokes DISTRIBUTEEVIDENCE on all adjacent cliques except C_i.

3.2. Messages

From the descriptions of the COLLECTEVIDENCE and DISTRIBUTEEVIDENCE algorithms, it is clear that evidence is passed between adjacent cliques by absorption. Consider the adjacent cliques C_i and C_j separated by S as shown in Fig. 3. Absorption (of evidence) from C_j to C_i over S amounts to computing a multiplicative decomposition Φ_S of the joint potential of S from the potentials associated with C_j and the neighbouring separators except S. Let $\mathcal{R}_S$ be the set of potentials associated with C_j and neighbours except S, then a multiplicative decomposition of ϕ_S is computed by elimination of all variables of $\mathcal{R}_S$ not in S. The absorption of evidence proceeds as:

Algorithm 3.3 (*Absorption*). Let C_i and C_j be adjacent cliques and let S be the separator between C_i and C_j. If *Absorption* is invoked on C_j from C_i, then:
(1) Set

$$\mathcal{R}_S = \Phi_{C_j} \cup \bigcup_{S' \in ne(C_j)\setminus\{S\}} \Phi_{S'}.$$

(2) **For** each variable X in $\{X \in dom(\phi) \mid \phi \in \mathcal{R}_S, X \notin S\}$
   (a) Marginalize out X.

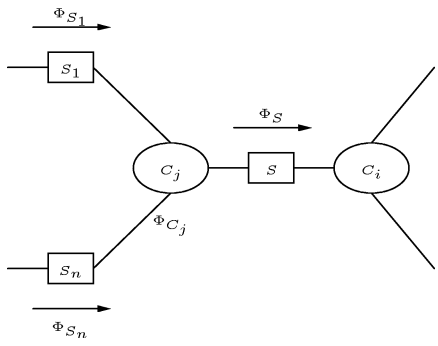

Fig. 3. Absorption from clique C_i to clique C_j over separator S.

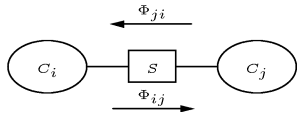

Fig. 4. None of the potentials passed from C_i to C_j are involved in any marginalizations when computing the message to pass in the opposite direction.

(3) Let Φ_S^* be the set of potentials obtained.
(4) Associate Φ_S^* with S as the set of potentials passed from C_j to C_i.

During message passing we want to avoid passing information received from an adjacent clique back to the same clique. The message passed from a clique C_i to an adjacent clique C_j contains all relevant information in the subtree rooted at C_i and not including C_j. The information contained in the message passed from C_i to C_j should not be contained in the message passed in the opposite direction (i.e., from C_j to C_i).

Consider the adjacent cliques C_i and C_j separated by S as shown in Fig. 4. Let Φ_{ij} be the set of potentials passed from C_i to C_j over S and let Φ_{ji} be the set of potentials passed in the opposite direction. None of the potentials in Φ_{ij} are involved in any marginalizations when computing the message Φ_{ji}. This implies that the division operation performed in HUGIN propagation quite simply amounts to discarding Φ_{ij} from Φ_{ji}.

The issue of avoiding to pass information received from a clique back to the same clique is solved differently in the HUGIN, Shafer–Shenoy, and LAZY propagation architectures. In the HUGIN architecture it is solved by division of separator potentials, in the Shafer–Shenoy architecture it is solved by computing the message to pass over S from the potentials associated with C_i and neighbouring separators except S, and in the LAZY propagation architecture it is solved by discarding the potentials passed to C_i over S from the set of potentials passed in the opposite direction. Thus, LAZY propagation dissolves the difference between HUGIN and Shafer–Shenoy propagation.

3.3. Evidence

To take advantage of the independence relations induced by evidence, it is necessary to modify the mechanism for incorporating evidence. Entering evidence to a single clique does not enable us to take full advantage of the independence relations induced by evidence. In the LAZY propagation architecture evidence is incorporated by associating evidence on a variable X with all cliques containing X. Soft evidence on a variable is, on the other hand, associated with a single clique of the junction tree.

Evidence on a variable $X = x$ is incorporated by reducing the domain of every potential containing X. If we think of a potential ϕ where $X \in dom(\phi)$ as represented by table t_ϕ, then all parts of t_ϕ not corresponding to $X = x$ are removed from t_ϕ. This implies that evidence reduces the size of the representation to include configurations of the domain where $X = x$ only. A function corresponding to performing this operation is referred to as an *evidence function*. Evidence on a variable X decreases the domain of each potential with X in its domain by X. The process of decreasing the representation of a potential ϕ to reflect the evidence will be referred to as *instantiation* of ϕ. The incorporation of evidence can be formalized as:

Algorithm 3.4 (*Entering of evidence*). If a variable X is observed to take on a value x (i.e., $X = x$), then instantiate all potentials ϕ where $X \in dom(\phi)$. If soft evidence on X is available, then associate a finding function with a clique containing X.

When evidence has been incorporated into the junction tree each clique has a list of evidence functions and a list of potentials associated. As evidence variables are instantiated, domains of potentials might be reduced to the empty set. Notice that potentials with empty domains are normalization constants.

3.4. Internal elimination

The basic computational structure of the LAZY propagation inference architecture is a junction tree constructed from the Bayesian network under consideration. The structure of the junction tree imposes a partial order on the set of elimination orders possible during inference. The separator S between two adjacent cliques C_i and C_j denotes the intersection of C_i and C_j. Therefore, it implicitly indicates which variables to eliminate when passing messages between C_i and C_j. The separator does not, however, indicate the order in which the variables should be eliminated. The process of eliminating a set of variables when computing a message to pass between two adjacent cliques will be referred to as *internal elimination of variables*.

The absorption algorithm (Algorithm 3.3) assumes that all potentials $\mathcal{R}_S$ associated with C_j and its neighbours except S are relevant for computing Φ_S^*. However, not all of the potentials in $\mathcal{R}_S$ are relevant for calculating the joint of S due to barren variables and independence relations induced by evidence. The set of relevant potentials $\mathcal{R}'_S$ can be determined in time linear in the size of the domain graph induced by $\mathcal{R}_S$ using a d-separation algorithm and the unity-potential axiom. If the potentials of $\mathcal{R}'_S$ contain variables which are not in S, then these variables are eliminated by marginalization. The

following algorithm can be used in step (1) of Algorithm 3.3 to determine the set of potentials relevant for computing Φ_S^*:

Algorithm 3.5 (*Find relevant potentials*). Let Φ be a set of potentials and let S be a set of variables. If *Find relevant potentials* for calculating the joint of S is invoked on Φ, then:

(1) Let $\mathcal{R}_S = \{\phi \in \Phi \mid \exists X \in dom(\phi)$ such that X is d-connected to $Y \in S\}$.
(2) Use the unity-potential axiom to remove from $\mathcal{R}_S$ all potentials containing only barren head variables to obtain $\mathcal{R}'_S$.
(3) Return $\mathcal{R}'_S$.

A message Φ_S computed by elimination of variables from $\mathcal{R}'_S$ does not contain evidence which is d-separated from the variables of S. If the entire set of evidence for some reason has to be included in the message Φ_S, then probabilities of the evidence d-separated from S has to be calculated. Notice that d-separation properties and barren variables can also be exploited to improve the efficiency of these additional computations.

3.5. Posterior marginals

In Section 2.5 we described how the posterior marginal of a variable is readily computed from a consistent junction tree in any of the standard inference architectures. Posterior marginals are also readily computed in the LAZY propagation architecture. The posterior distribution of a variable X can be computed from any clique or separator containing X. As clique and separator potentials are factorized multiplicatively computing a marginal may involve combination of potentials as well as elimination of variables. It is straightforward to see that the tasks performed when calculating a posterior marginal are precisely the tasks performed when calculating a message. The algorithms for finding the relevant potentials and marginalization can be used to calculate a multiplicative representation of the posterior marginal of a variable. Now, all that remains is to combine the potentials and thereby obtain the posterior marginal:

Algorithm 3.6 (*Posterior marginal*). Let Φ be the set of potentials representing the joint distribution from which the posterior marginal of Y is to be calculated. The posterior marginal $P(Y|\varepsilon)$ is calculated by performing the following steps:

(1) Invoke find relevant potentials on Φ to obtain $\mathcal{R}_Y$.
(2) **For** each variable X in $\{X \in dom(\phi) \mid \phi \in \mathcal{R}_Y, X \neq Y\}$
    (a) Marginalize out X.
(3) Let Φ_Y be the set of potentials obtained.
(4) Calculate

$$P(Y|\varepsilon) = \frac{\prod_{\phi\in\Phi_Y} \phi}{\sum_Y \prod_{\phi\in\Phi_Y} \phi}.$$

3.6. Correctness of LAZY propagation

The LAZY propagation architecture is based on the representations and algorithms described in the previous sections. The architecture would of course be useless, if the calculations performed did not produce the correct results:

Theorem 3.1 (LAZY propagation). *Let C be a clique, let S be a neighbouring separator, and let $\varepsilon = \{\varepsilon_1, \ldots, \varepsilon_n\}$ be the evidence. After a full round of message passing, we have*:

$$P(C|\varepsilon) \propto \prod_{\phi \in \Phi_C} \phi \prod_{i=1}^{n} \varepsilon_i \prod_{S' \in ne(C)} \prod_{\phi' \in \Phi_{S' \to C}} \phi',$$

$$P(S|\varepsilon) \propto \prod_{\phi \in \Phi_{S \to C}} \phi \prod_{\phi' \in \Phi_{S \leftarrow C}} \phi',$$

where Φ_C is the set of potentials associated with C, $\Phi_{S' \to C}$ is the set of potentials passed to C over S', and $\Phi_{S \to C}$ and $\Phi_{S \leftarrow C}$ are the sets of potentials passed over S.

Proof. We will argue for the correctness of the LAZY propagation architecture described in Sections 3.1–3.3. The basic idea of LAZY propagation is to maintain decompositions of clique and separator potentials. Message passing in the LAZY propagation architecture proceeds as in the Shafer–Shenoy architecture. The potentials initially associated with each clique are, however, not combined to form the initial clique potential and a message passed over a separator contains a set of potentials. Let Φ be the set of potentials associated with a clique C and all potentials passed to C (or the union of the sets of potentials passed over a separator S) of a consistent junction tree in the LAZY propagation architecture. The set Φ corresponds to the potential(s) associated with C (or S) in the Shafer–Shenoy architecture. This implies that there is nothing to prove as the considerations with respect to internal elimination described in Section 3.4 do not effect the correctness of the architecture. □

3.7. Examples

Before proceeding with the description of the LAZY propagation architecture we present the following two examples.

Example 3.2 (LAZY *propagation*). Consider the Bayesian network $N = (G, \mathcal{P})$ shown in Fig. 5 and the corresponding (suboptimal) junction tree $\mathcal{T}$ shown in Fig. 6. The junction tree $\mathcal{T}$ is not initialized, but the prior probability distributions of $\mathcal{P}$ are associated with cliques as indicated in the figure. We will assume that the evidence is $\varepsilon = \{D = d\}$. This implies that each potential with D in its domain has its domain decreased by D.

Let clique *BCDEF* be predetermined as the root of the junction tree. The junction tree is made consistent by invoking COLLECTEVIDENCE and DISTRIBUTEEVIDENCE on *BCDEF*. When COLLECTEVIDENCE is invoked on *BCDEF*, messages will flow from the leaf cliques to *BCDEF* as depicted in Fig. 7.

The set of potentials associated with clique *ABC* is $\{P(A), P(B|A), P(C|A)\}$. The variable A has to be eliminated from this set of potentials. A is eliminated by

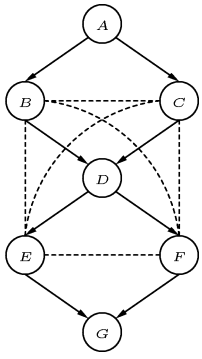

Fig. 5. A Bayesian network. Dashed lines are fill-ins added during moralization and triangulation of the network.

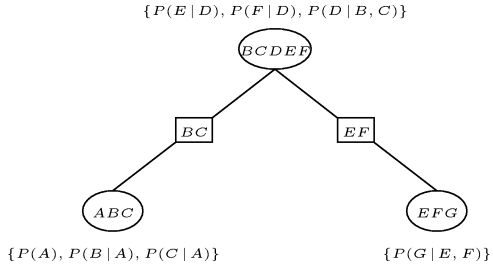

Fig. 6. A junction tree constructed from the Bayesian network shown in Fig. 5. Prior distributions are associated with cliques as indicated.

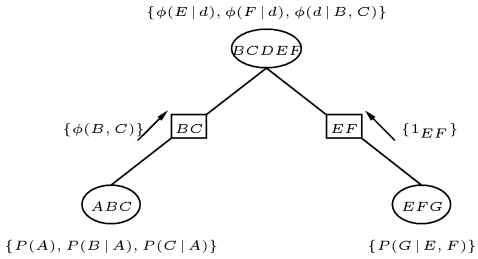

Fig. 7. Shows the message flow in the junction tree during COLLECTEVIDENCE when D is instantiated to d.

marginalization over the combination of $P(A)$, $P(B|A)$, and $P(C|A)$. The resulting potential has domain $\{B, C\}$:

$$\phi(B, C) = \sum_A P(B|A)P(C|A)P(A).$$

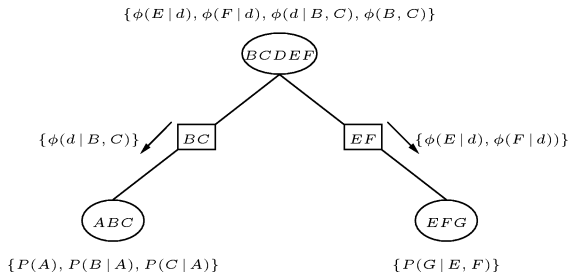

Fig. 8. Shows the message flow in the junction tree during DISTRIBUTEEVIDENCE when D is instantiated to d.

The set of potentials associated with clique EFG is $\{P(G \mid E, F)\}$. Variable G has to be eliminated, but no calculations are needed as G is a barren variable and the unity-potential axiom applies, i.e.:

$$\phi(\cdot \mid E, F) = \sum_G P(G \mid E, F) = 1_{E,F}.$$

When DISTRIBUTEEVIDENCE is invoked on *BCDEF*, messages will flow from *BCDEF* to the leaf cliques as depicted in Fig. 8.

The set of potentials associated with *BCDEF* after COLLECTEVIDENCE is $\{\phi(E \mid d), \phi(F \mid d), \phi(d \mid B, C), \phi(B, C)\}$. The set of potentials relevant for computing the message to pass to *ABC* is $\{\phi(B, C), \phi(d \mid B, C)\}$. $\phi(B, C)$ was passed in the opposite direction during COLLECTEVIDENCE and the domain of $\phi(d \mid B, C)$ is equal to the domain of the separator. Hence, no computations are needed to obtain the message to pass to *ABC*.

The set of potentials relevant for computing the message to pass to *EFG* is $\{\phi(E \mid d), \phi(F \mid d)\}$. Again, no computations are needed as the domains of the relevant potentials are subsets of the separator.

The example shows how LAZY propagation exploits the independence relations induced by the evidence on D to reduce the computational cost of inference. The evidence on D d-separates $\{A, B, C\}$ from $\{E, F, G\}$.

The evidence on D has been propagated in $\mathcal{T}$ by performing only a single marginalization and two combinations of potentials. This is much less than what is required in both HUGIN and Shafer–Shenoy propagation. Notice that even though the triangulation was suboptimal, the efficiency of the LAZY propagation algorithm was not effected. In the example LAZY propagation alleviates the sub-optimality of the triangulation.

During internal elimination of variables LAZY propagation is able to take advantage of independence relations induced by evidence. This is illustrated further in the following example.

Example 3.3 (*Exploiting d-separation*). Consider the Bayesian network $N = (G, \mathcal{P})$ shown in Fig. 9 and the corresponding junction tree $\mathcal{T}$ shown in Fig. 10.

We will describe how independence relations induced by different sets of evidence effect the computational efficiency of calculating the posterior marginal of E.

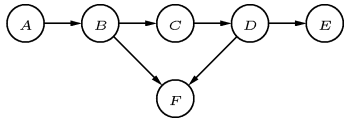

Fig. 9. A Bayesian network G where A and E are independent given C.

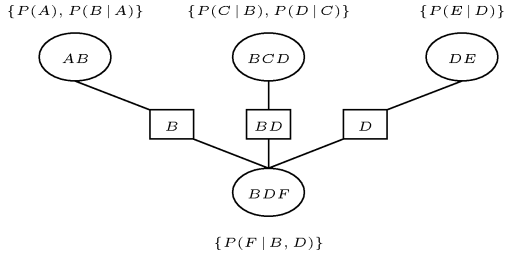

Fig. 10. A junction tree for N. The potentials of N are associated with the cliques as indicated.

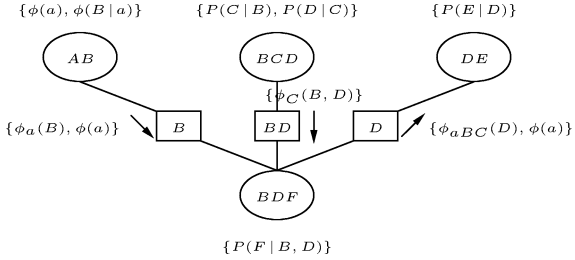

Fig. 11. The flow of messages to clique DE when A is instantiated to a.

Initially, A and E are not d-separated. If C is instantiated, then A and E are d-separated, but if both C and F are instantiated, then A and E are not d-separated.

If we assume that A is instantiated to a, then Fig. 11 illustrates the flow of messages towards clique DE. The index of a potential indicates the variables and the evidence relevant for the calculation of the corresponding potential.

Fig. 11 indicates that the evidence $\{A = a\}$ and variables B and C (and D) are relevant for computing the posterior marginal of E. F, on the other hand, is irrelevant. This has reduced the computational cost of calculating the posterior marginal of E as the cost of determining that F is a barren variable is negligible. The cost of propagating evidence to DE is close to the cost of propagating evidence in a junction tree for $G = (V \setminus \{F\}, E \setminus \{(B, F), (D, F)\})$.

Next, assume also that C is instantiated to c. The flow of messages towards clique DE is illustrated in Fig. 12. We see that only $C = c$ is relevant for E, and the fact that E and A are d-separated has yield substantial reductions in the computational cost. No marginalizations are performed during the propagation of evidence.

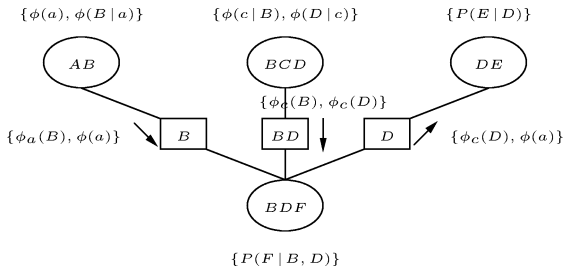

Fig. 12. The flow of messages to clique *DE* when A is instantiated to a and C to c.

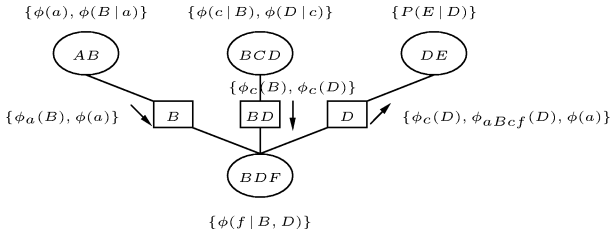

Fig. 13. The flow of messages to clique *DE* when A is instantiated to a, C to c, and F to f.

For completeness we also illustrate the flow of messages when F is instantiated to f. Evidence on F d-connects A and E and the computational cost of computing the posterior of E increases, see Fig. 13.

3.8. Sample empirical results

Proposing yet another architecture for performing probabilistic inference in Bayesian networks seems unnecessary unless the proposed architecture offers improvements over the standard inference architectures. We have measured the performance of LAZY propagation relative to Shafer–Shenoy and HUGIN propagation in junction trees with respect to time and space usage.

The performance of LAZY propagation has been tested with respect to the time used for both inference and calculation of all posterior marginals, the size of the largest potential computed during inference, and the total size of the initialized junction tree.

The performance tests were performed on a number of large real-world Bayesian networks. The tests were performed with evidence sets of different sizes. For a given Bayesian network and a fixed number of evidence variables 25 propagations were performed with randomly selected evidence variables. The number of evidence variables varied from 0 to 75. Tables 1 and 2 describe the structure of three of the Bayesian networks and corresponding junction tree representations used in the tests.

Figs. 14–16 show plots of the average time for propagation of evidence as a function of the size of the randomly selected evidence sets for the Diabetes, KK-KVL-maltbyg (KK), and ship-ship networks, respectively.

Table 1
Information on three of the Bayesian networks used in the tests

Network	Nodes	Node potential size		
		Min	Max	μ
Diabetes	413	5	7056	1116.4
KK	50	2	32256	2768.1
Ship-ship	50	2	18270	2609.6

Table 2
Information on the junction trees corresponding to the Bayesian networks of Table 1.

Network	Cliques	Clique state space size			Clique neighbours		
		Min	Max	μ	Min	Max	μ
Diabetes	337	495	190080	30906.3	1	3	2.0
KK	38	40	5806080	397780.2	1	4	1.9
Ship-ship	35	8	4032000	693102.1	1	3	1.9

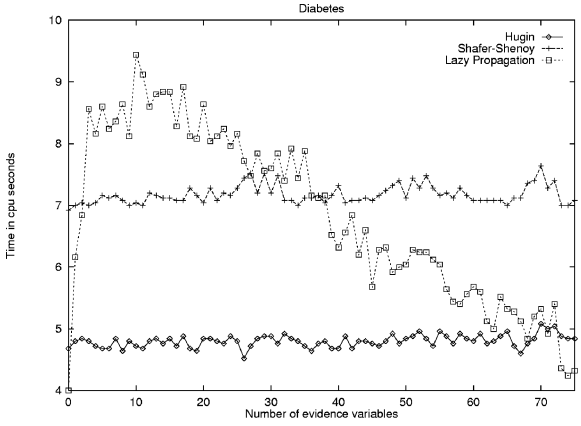

Fig. 14. A plot of the average time cost of propagating evidence in the Diabetes network as a function of the number of variables instantiated.

The plots clearly show that the time cost of LAZY propagation decreases as the number of instantiated variables increase. Similar performance results were obtained for all the Bayesian networks used in the tests. Figs. 15 and 16 show two plots where the time cost of LAZY propagation is lower than the time cost of both HUGIN and Shafer–Shenoy propagation even when no variables are instantiated. Fig. 14 shows a plot where the time cost of LAZY propagation is higher than the time costs of Shafer–Shenoy and HUGIN propagation for small sets of evidence. The time cost of LAZY propagation decreases as the size of the set of evidence reaches ten variables. For all but a few of the tests performed the

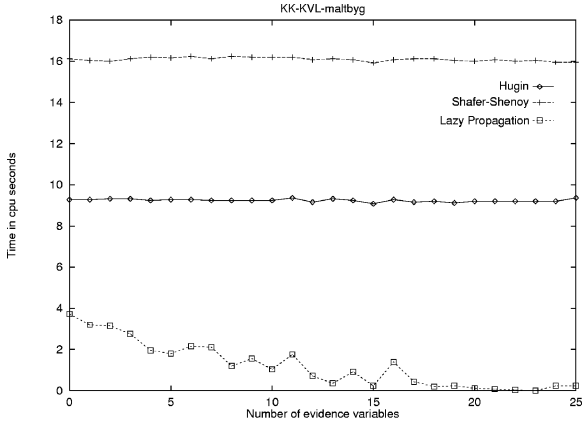

Fig. 15. A plot of the average time cost of propagating evidence in the KK network as a function of the number of variables instantiated.

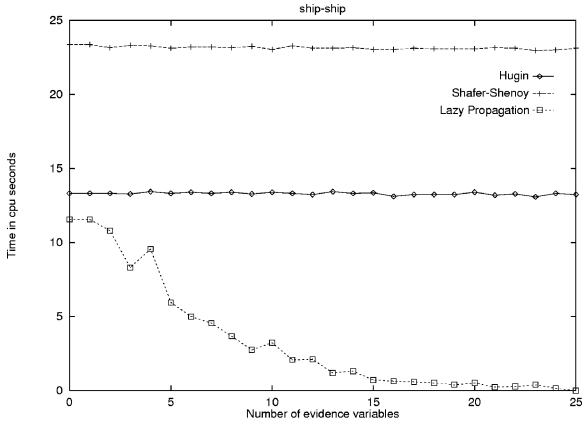

Fig. 16. A plot of the average time cost of propagating evidence in the ship-ship network as a function of the number of variables instantiated.

time cost of LAZY propagation is lower than the time cost of Shafer–Shenoy and HUGIN propagation for all sizes of the set of evidence variables.

Plots of the average size (in terms of numbers) of the initialized junction tree and the largest potential (in terms of numbers) computed during inference are shown in Appendices A and B, respectively.

The tests of the space cost indicate that the size of the initialized junction tree in the LAZY propagation architecture is substantially smaller than the initialized junction trees in both the Shafer–Shenoy and HUGIN architectures. The tests also indicate that the size of the initialized junction tree in the LAZY propagation architecture decreases as the size of the evidence set increases. Due to exploitation of the d-separation criteria the decrease

in size is often larger than the decrease obtained when potential domains are reduced by instantiated evidence variables.

The plots of the average size of the largest potential computed during inference clearly indicate that the size of the largest potential computed in the LAZY propagation architecture decreases as the size of the evidence set increases. The plots of the average size of the largest potential show that the average size of the largest potential is the same for the three architectures when the set of evidence is empty. This was, however, not always the case. The average size of the largest potential in the LAZY propagation architecture was sometimes smaller than the average size of the largest potential in the other two architectures when the evidence set was empty.

It should be noted that the tests were designed to measure the performance of LAZY propagation relative to Shafer–Shenoy and HUGIN propagation in junction trees. The tests were not designed to measure the relative performance of Shafer–Shenoy and HUGIN propagation. For example, we have not used binary join trees as the computational structure of Shafer–Shenoy propagation [28].

3.9. Various speed-up improvements

There exists a lot of different heuristics for speeding up inference in standard junction tree architectures. The sample empirical results reported in the previous section are all based on implementations which exploit only very few improvements to speed up the inference task.

The only speedup improvement implemented is related to the use of the unity-potential axiom in the LAZY propagation architecture. The improvement is referred to as *expression trees*. Expression trees are used in order to be able to recognize situations where elimination of a set of variables will lead to a unity potential. Consider the junction tree depicted in Fig. 17 (this junction tree is the result of a poor triangulation). The main point to notice is that it should be recognized that the elimination of variables A and B from the potential $\phi(A, B \mid D)$ when calculating the message to pass from $ABDE$ to EF produces a unity potential.

This is a very simple example which can be recognized without the use of expression trees, but the situation might be much more involved and therefore expression trees are necessary.

A number of different improvements to standard junction tree propagation architectures have been presented through the years. Zero compression [12], binary join trees [28] and nested junction trees [17] (which can be regarded as a special case of LAZY propagation)

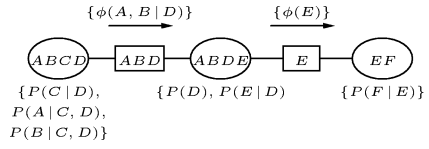

Fig. 17. When calculating the message to pass from $ABDE$ to EF, it should be recognized that elimination of A and B before elimination of D will produce a unity potential.

are just a few of them. A number of improvements to the Shafer–Shenoy and HUGIN architectures are described in [24].

Other improvements involve cache considerations, optimal factorings of combination of potentials, and non-myopic heuristic triangulation methods. We will not consider any of the improvements in this paper, but just mention that all the improvements of the standard junction tree inference architectures known to the authors are also applicable to the LAZY propagation inference architecture. The impact of the improvements on the performance of LAZY propagation is a subject of future research.

4. Solving common Bayesian network tasks

In this section we describe the impact of LAZY propagation on a set of commonly performed Bayesian network tasks. The tasks are calculation of joint probabilities, cautious propagation of evidence, determining a most probable configuration, fast retraction of evidence, and handling evidence arriving incrementally. Finally, we also consider how independence of causal influence can be exploited in the LAZY propagation architecture to reduce both the time and space cost of inference.

4.1. Joint probability

It is not always sufficient to have access to the marginal distribution of each variable in the Bayesian network. It may be important to compute the joint probability distribution of an arbitrary subset of the variables. If the subset of variables V of interest is known prior to the construction of the junction tree, then the variables of V can be forced into the same clique C by adding fill-in edges between all the variables of V. The joint marginal of V is then computable from the potential of C. If V is a subset of a clique C, then it is straightforward to calculate the joint probability distribution of V:

$$P(V) = \sum_{C \setminus V} \prod_{\phi \in \Phi_C} \phi \prod_{S \in ne(C)} \prod_{\phi' \in \Phi_S} \phi',$$

where Φ_C is the set of potentials associated with C and Φ_S is the received potentials associated with each neighbouring separator S. In the remaining part of this paper we will only consider the case where V is not known prior to the construction of the junction tree.

If V is not a subset of any of the cliques in the junction tree, then there exists a number of different approaches to calculating the joint probability distribution of V. Variable propagation and variable firing are simple methods which are readily exploited by any junction tree propagation algorithm, see, e.g., [14].

With variable propagation the joint probability distribution of any subset of variables V is calculated by passing message to a clique C. The clique C can be chosen as the clique containing the largest subset of variables of V. The message passing proceeds such that none of the variables of V are eliminated. This gives easy access to the joint distribution of V. The joint can be obtained from the clique potential of C by eliminating all variables Y where $Y \in C$ and $Y \notin V$. Variable propagation in standard junction tree inference architectures requires some additional control structures as the variables propagated have to

be included in all cliques and separators on the path between C and the cliques containing variables of the joint. Computing a joint distribution of a set of variables V in the LAZY propagation architecture by propagation of variables is straightforward as it just amounts to not eliminating the variables of V each time a message is computed. The modified absorption algorithm is:

Algorithm 4.1 (*Joint absorption*). Let C_i and C_j be adjacent cliques, let S be the separator between C_i and C_j, and let V be the desired joint. If *Absorption* is invoked on C_j from C_i, then:

(1) Set

$$\mathcal{R}_S = \Phi_{C_j} \cup \bigcup_{S' \in ne(C_j)\setminus\{S\}} \Phi_{S'}.$$

(2) **For** each variable X in $\{X \in dom(\phi) \mid \phi \in \mathcal{R}_S, X \notin S, X \notin V\}$
   (a) Marginalize out X.
(3) Let Φ_S^* be the set of potentials obtained.
(4) Associate Φ_S^* with S as the set of potentials passed from C_j to C_i.

Xu [30] introduces another method for computing the joint distribution of a set of variables V. The method is based on building a second layer junction tree on top of the existing junction tree. The second layer junction tree is build from a minimal connected subtree of the original junction tree containing all variables of V. The second layer junction tree is constructed such that the root of the tree contains all variables of V. When the second layer junction tree has been constructed, the probability distribution of V can be calculated by collecting evidence to the root of the second layer junction tree. The method is independent of the particular inference algorithm used. This implies that LAZY propagation probably will have reduced computational cost compared to standard junction tree propagation algorithms.

The goal of the method proposed by Xu is to construct a junction tree with the joint of interest located in the root of the second layer junction tree. If we are not interested in the joint probability distribution, but only in a representation of the joint distribution and if the joint distribution decomposes multiplicatively, then we can construct the second layer junction tree such that we have a set of cliques only containing variables of the joint. This can reduce the space cost of representing the joint.

4.2. Cautious propagation

A Bayesian network $N = (G = (V, E), \mathcal{P})$ is a closed world representation of a joint probability distribution of V. Given a set of evidence ε on a subset of V, we would like to detect any conflicts in ε and to detect any inconsistencies between ε and the model. This is referred to as *conflict analysis*. The main task involved in conflict analysis is calculating $P(\varepsilon')$ for each $\varepsilon' \subseteq \varepsilon$.

Furthermore, we also want to determine how sensitive a set of hypotheses $\mathcal{H} = \{h_1, \ldots, h_n\}$ is to changes in ε. This is referred to as *sensitivity analysis*. The main task involved in sensitivity analysis is calculating $P(h|\varepsilon')$ for each $\varepsilon' \subseteq \varepsilon$ and $h \in \mathcal{H}$.

The number of subsets $\varepsilon' \subseteq \varepsilon$ grows exponentially as the number of findings increases. Hence, the task of computing $P(\varepsilon')$ for all subsets ε' and $P(h|\varepsilon')$ for all subsets ε' and all hypotheses h might become an intractable task to perform with standard inference algorithms. The task is especially heavy when $P(\varepsilon')$ or $P(h \mid \varepsilon')$ have to be calculated through a propagation in a large junction tree. Cautious propagation [13] offers the opportunity to compute the probability of a large number of these probabilities from a single propagation. The prize paid is a small decrease in time and storage efficiency.

Cautious propagation is an inference algorithm based on message passing in a junction tree representation. To compute $P(\varepsilon')$ for different subsets ε' of the evidence ε, the junction tree is first made consistent with no evidence inserted. The initial propagation can be performed with any junction tree based inference algorithm. We will assume that the HUGIN inference algorithm is used to perform the initial propagation. The main idea of cautious propagation is to change the content of the messages passed over separators in the HUGIN architecture. The separator messages are changed from containing $\phi(S, \varepsilon)$ to containing $\phi(\varepsilon \mid S)$. When a clique C receives a message from an adjacent clique via a separator S, the clique potential ϕ_C is not updated. Instead the message is associated with S implying that each separator contains three potentials. That is, each separator S contains the initial joint potential of S along with the two potentials passed over S during cautious propagation. To facilitate easy access to complements of individual pieces of evidence, evidence is entered as finding potentials. The finding potentials associated with a clique are treated in the same way as messages from neighbour separators. This is by Jensen [13] referred to as *cautious entering of evidence*.

4.2.1. Cautious messages

Assume that C_i and C_j are adjacent cliques separated by S. Fig. 18 shows the situation before evidence is cautiously propagated from C_j to C_i. C_j has received messages from all adjacent cliques except possibly C_i. The finding potentials associated with C_j are indicated as a list of potentials $\varepsilon_1, \ldots, \varepsilon_m$.

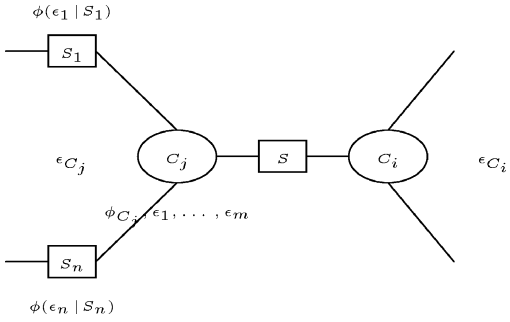

Fig. 18. The situation before evidence is cautiously propagated from clique C_j to clique C_i.

The message passed from C_j to C_i is the conditional probability potential $\phi(\varepsilon_{C_j} \mid S)$ of the evidence ε_{C_j} entered to the subtree rooted at C_j and not including C_i given the variables of S. This message is easily calculated based on the calculation of $\phi(\varepsilon_{C_j}, S)$:

$$\phi(\varepsilon_{C_j}, S) = \sum_{C_j \setminus S} \phi_{C_j} \prod_{i=1}^{n} \phi(\varepsilon_i \mid S_i) \prod_{k=1}^{m} \varepsilon_k,$$

where $\varepsilon_{C_j} = \bigcup_{i=1}^{n} \varepsilon_i \cup \bigcup_{k=1}^{m} \varepsilon_k$. Cautious propagation proceeds as HUGIN propagation except for the changes described above. The division performed in HUGIN propagation produces the potential $\phi(\varepsilon_{C_j} \mid S)$.

4.2.2. Probabilities of sets of evidence

Performing cautious propagation of evidence in a junction tree does not make it possible to compute $\phi(\varepsilon')$ for all subsets of the evidence $\varepsilon' \subseteq \varepsilon$. Each clique C_i receives a set of messages from its adjacent cliques. If $\phi(\varepsilon_{C_j} \mid S)$ is a message received by C_i, then ε_{C_j} is the subset of evidence incorporated into the subtree of the junction tree rooted at C_j and not including C_i. From the messages received by C_i and the evidence cautiously entered at C_i probabilities of subsets of the evidence can be calculated. The evidence ε_{C_j} is contained in the potential $\phi(\varepsilon_{C_j} \mid S)$ and it is therefore not possible at C_i to partition ε_{C_j} into subsets. This implies that the subsets of evidence for which probabilities can be computed at C_i are combinations of the subsets of evidence received by C_i and the evidence cautiously entered at C_i. In the rest of this paper we use the notation $\varepsilon' \subseteq \varepsilon$ to refer to the subsets of evidence for which probabilities can be computed.

The joint probability of any subset of the evidence received at clique C and the evidence associated with C can easily be computed from the messages received by C, the clique potential of C, and the set of finding potentials associated with C:

Example 4.1. The joint probability of evidence subsets ε_i and ε_j passed over separators S_i and S_j can be computed in the following way:

$$\phi(\varepsilon_i, \varepsilon_j) = \sum_{C} \phi(\varepsilon_i \mid S_i)\phi(\varepsilon_j \mid S_j)\phi(C), \tag{2}$$

where the clique potential $\phi(C)$ contains no evidence.

Notice that because each piece of evidence is associated with a single clique, it is only possible to retract each piece of evidence from the corresponding clique.

4.3. LAZY *cautious-propagation*

Cautious propagation is concerned with computing probabilities of subsets of the evidence. In order to give easy access to as many subsets of the evidence as possible, a number of small adjustments is made to the LAZY propagation architecture. Even though the adjustments are simple, we will refer to LAZY propagation extended with the adjustments as LAZY *cautious-propagation*.

Cautious propagation is performed in a junction tree initialized with no evidence. During cautious propagation each message passed over a separator S is stored at S along with the

joint potential of S computed during the initial propagation. In the LAZY propagation architecture the initial potential for a separator S is not constructed explicitly. Instead two sets of potentials are associated with S. Thus, in the LAZY cautious-propagation architecture four sets of potentials are associated with each separator.

The main idea of propagating evidence cautiously is to propagate the entire set of evidence in a way such that it is easy to retract different subsets of the evidence. Thus, evidence is retracted, but never added in the process of computing probabilities of subsets of the evidence. Therefore, if a variable X is a barren variable when propagating the entire set of evidence, then X will also be a barren variable when some evidence is retracted. This is formalized in the following proposition:

Proposition 4.2. *Evidence can never make a variable barren.*

Proof. Follows immediately from the definition of barren variables. □

The proposition implies that the unity-potential axiom can be exploited as usual during LAZY cautious-propagation.

In order to make it possible to facilitate retraction of evidence, evidence on variables should not be incorporated by instantiation of potentials right away. Let X be a variable instantiated by evidence and let ε_X be the corresponding evidence function. The potentials containing X should not be instantiated right away as this will make it impossible to retract ε_X. Instead the evidence function ε_X is associated with each clique containing X. The instantiation of X is postponed until the point where X would be eliminated, if it had not been instantiated by evidence. This implies that the opportunity to exploit the independence relations induced by ε_X is not reduced considerably. Furthermore, this also makes it possible to retract ε_X from more than one clique. The algorithm for entering evidence has to be changed slightly such that domains of potentials including instantiated evidence variables are not decreased right away:

Algorithm 4.2 (LAZY *cautious entering of evidence*). If a variable X is observed to take on a value x, then associate an evidence function with all cliques C with $X \in C$. If soft evidence on X is available, then associate a finding function with a clique containing X.

Notice that Algorithm 4.2 is similar to Algorithm 3.4.

4.3.1. Message passing

With the above described adjustments in mind we now proceed to describe how the message passing phase of LAZY cautious-propagation proceeds. Messages passed between cliques of the junction tree consist of sets of potentials as described in Section 3. The set of potentials Φ_S passed from a clique C_j to an adjacent clique C_i over a separator S can be calculated as usual except for the special notice given to evidence functions (see Fig. 19):

Algorithm 4.3 (*Cautious absorption*). Let C_i and C_j be adjacent cliques, let S be the separator between C_i and C_j, and let ε_{C_j} be the evidence associated with C_j. If *Absorption* is invoked on C_j from C_i, then:

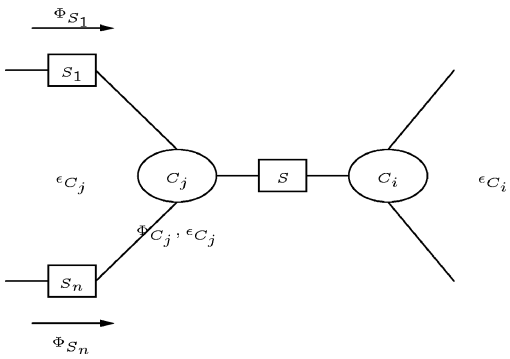

Fig. 19. LAZY cautious-propagation of evidence.

(1) Set

$$\mathcal{R}_S = \Phi_{C_j} \cup \varepsilon_{C_j} \cup \bigcup_{S' \in ne(C_j)\setminus\{S\}} \Phi_{S'}.$$

(2) **For** each variable X in $\{X \in dom(\varepsilon) \mid \varepsilon \in \mathcal{R}_S, X \notin S\}$
   (a) Instantiate X.
(3) Let $\mathcal{R}_S^*$ be the set of potentials obtained.
(4) **For** each variable X in $\{X \in dom(\phi) \mid \phi \in \mathcal{R}_S^*, X \notin S\}$
   (a) Marginalize out X.
(5) Let Φ_S^* be the set of potentials obtained.
(6) Associate Φ_S^* with S as the set of potentials passed from C_j to C_i.

Algorithm 4.3 is similar to Algorithm 3.3. The only difference is that evidence is entered as evidence functions in the LAZY cautious-propagation architecture in order to facilitate retraction. The message passing phase of LAZY cautious-propagation proceeds in exactly the same way as message passing in LAZY propagation.

4.3.2. Probabilities of sets of evidence

It should be noticed that except for the simple adjustments mentioned above, LAZY cautious-propagation is equivalent to LAZY propagation. Therefore, the concept of cautious propagation more or less disappears when switching from HUGIN propagation (or Shafer–Shenoy propagation) to LAZY propagation as the underlying propagation algorithm. Only a change in the application of the algorithms for absorption and entering evidence is required to perform LAZY cautious-propagation in the LAZY propagation architecture.

Example 4.3. The joint probability distribution of the evidence ε and the variables of a clique C can be calculated as the product of the potentials associated with C and the received potentials associated with neighbouring separators:

$$P(C,\varepsilon) = \prod_{\phi\in\Phi_C} \phi \prod_{\varepsilon\in\varepsilon_C} \varepsilon \prod_{\phi'\in\bigcup_{S\in ne(C)}\Phi_S} \phi',$$

where ε_C is the set of evidence functions associated with C.

Example 4.4. The marginal distribution of the evidence ε' passed over a separator S from C_i to C_j is calculated as:

$$P(\varepsilon') = \sum_S \prod_{\phi\in\Phi_{C_j\to C_i}} \phi \prod_{\phi'\in\Phi_{C_i\to C_j}} \phi',$$

where $\Phi_{C_j\to C_i}$ is the message containing no evidence passed from C_j to C_i. Notice that the evidence ε' is contained in the set of potentials $\Phi_{C_i\to C_j}$.

Let ε_i and ε_j be the evidence passed to a clique C over two different separators S_i and S_j in the cautious propagation architecture. Eq. (2) shows how to compute the joint of ε_i and ε_j. This probability is equally simple to compute in the LAZY cautious-propagation architecture:

$$P(\varepsilon_i,\varepsilon_j) = \sum_C \prod_{\phi\in\Phi_C} \phi \prod_{\phi_i\in\Phi_{S_i}} \phi_i \prod_{\phi_j\in\Phi_{S_j}} \phi_j \prod_{S\in ne(C)\setminus\{S_i,S_j\}} \prod_{\phi'\in\Phi_S} \phi',$$

where Φ_S for each $S \in ne(C) \setminus \{S_i, S_j\}$ and Φ_C are the sets of potentials containing no evidence.

LAZY cautious-propagation can give access to a larger number of subsets of evidence than cautious propagation. Cautious propagation propagates $\phi(\varepsilon \mid S)$ potentials over the separators of the junction tree while LAZY propagation propagates multiplicative decompositions of $\phi(\varepsilon, S' \mid S'')$ where $S', S'' \subseteq S$. For example, if two pieces of evidence passed over a separator are independent, then these two pieces of evidence will be represented in two separate potentials in the message. In cautious propagation only one potential is passed over each separator, so the independence relations will not be exploited to make a larger number of subsets of the evidence accessible. The following example shows how LAZY cautious-propagation gives access to the probability of a larger number of subsets of the evidence.

Example 4.5. Consider the Bayesian network $N = (G, \mathcal{P})$ shown in Fig. 20. Assume that the evidence is $\varepsilon = \{A = a, D = d, F = f\}$.

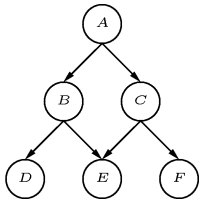

Fig. 20. A Bayesian network where LAZY cautious-propagation gives access to the probability of a larger number of subsets of the evidence than cautious propagation.

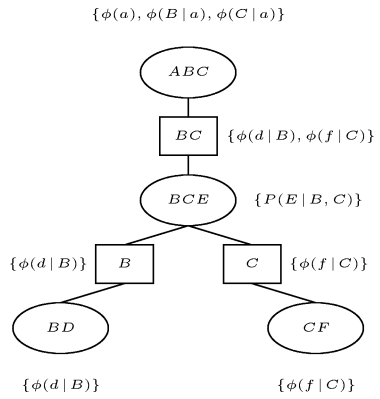

Fig. 21. LAZY cautious-propagation can exploit that the evidence $D = d$ and $F = f$ is d-separated by the evidence $A = a$.

Fig. 21 shows a junction tree representation of N and the flow of messages during COLLECTEVIDENCE in LAZY cautious-propagation. At the root clique it is possible to compute both $P(a, d)$ and $P(a, f)$. This is not possible using cautious propagation. It is, however, possible to compute these probabilities at clique *BCF* using cautious propagation, but this is not important for the example.

4.4. Max-propagation

A *most probable configuration* (also called a *most probable explanation*) of the variables V of a Bayesian network $N = (G, \mathcal{P})$ is defined to be a configuration of V with highest probability. A method for calculating a most probable configuration is described by Dawid [7]. The method is based on performing a propagation of evidence where variables are eliminated by maximization followed by a distribution of configurations of maximal probability. This is referred to as *max-propagation*. Thus, a max-propagation proceeds like a sum-propagation except that the marginalization operator is maximization. The definition of a *max-consistent junction tree* is equivalent to the definition of a consistent junction tree.

When the junction tree is max-consistent, a most probable configuration can be determined. If exactly one configuration of maximal probability exists, then the most probable configuration of V can be determined by combining the most probable configurations of each clique potential. If, on the other hand, more than one configuration with maximal probability exists, then a most probable configuration can be determined by performing a *max-configuration distribution*. Starting from the root clique R of the junction tree a most probable configuration $\widehat{r}$ is determined. Assume C is an adjacent clique of R and that the two cliques are connected by the separator S. A configuration $\widehat{s}$ of the separator variables is determined from $\widehat{r}$ and distributed to C. A most probable configuration $\widehat{c}$ of the variables in C is determined as the maximizing arguments of the clique potential ϕ_C with the variables of S instantiated to $\widehat{s}$:

$$\widehat{c} = \arg\max_{C \setminus S} \phi_C(C \setminus S, \widehat{s}).$$

Max-configuration distribution proceeds to the leaves of the junction tree as described above. After termination, a most probable configuration of the variables of the junction tree has been determined.

4.5. LAZY *max-propagation*

The only difference between sum- and max-propagation in standard junction tree inference architectures is that maximization is used instead of summation as the marginalization operator in max-propagation. Changing the marginalization operator from summation to maximization implies that the concept of barren variables vanishes:

$$\max_H P(H|T) \neq 1_T.$$

This again implies that the some of the efficiency improvements of LAZY propagation described in Section 3.9 disappear. For ease of reference we will refer to the max-propagation algorithm based on lazy evaluation as LAZY *max-propagation*. The LAZY max-propagation algorithm starts out from the same initial junction tree representation as LAZY propagation.

4.5.1. Message passing

With the above described adjustment in mind we now proceed to describe how the message passing phase of LAZY max-propagation proceeds. Messages passed between cliques in the junction tree consist of sets of potentials as described in Section 3. The set of potentials Φ_S passed from clique C_i to a neighbouring clique C_j over a separator S can be calculated as usual. The algorithm for marginalization is, however, changed slightly:

Algorithm 4.4 (*Max marginalization*). Let $\Phi = \{\phi_1, \ldots, \phi_n\}$ be a set of potentials. If *Max marginalization* of X is invoked on Φ, then:

(1) Set $\Phi_X = \{\phi \in \Phi \mid X \in dom(\phi)\}$.

(2) Calculate

$$\phi_X^* = \max_X \prod_{\phi \in \Phi_X} \phi.$$

(3) Let $\Phi^* = \{\phi_X^*\} \cup \Phi \setminus \Phi_X$.

Φ^* is the resulting set of potentials when X is eliminated from Φ.

Notice that only the marginalization operation is changed relative to Algorithm 2.3.

Let Φ be the set of potentials associated with C and its neighbours except S. The set of potentials $\mathcal{R}_S$ relevant for computing the message Φ_S to pass over S can be determined by running a d-separation algorithm on Φ. The algorithm for finding the set of relevant potentials for computing Φ_S is:

Algorithm 4.5 (*Find max relevant potentials*). Let Φ be a set of potentials and let S be a set of variables. If *Find max relevant potentials* for calculating the joint of S is invoked on Φ, then:

(1) Let $\mathcal{R}_S = \{\phi \in \Phi \mid \exists X \in dom(\phi)$ such that X is d-connected to $Y \in S\}$.
(2) Return $\mathcal{R}_S$.

The only difference between Algorithms 3.5 and 4.5 is that the step to remove barren variables and their potentials does not apply to LAZY max-propagation.

4.5.2. Determine max configuration

After a full round of message passing in the junction tree using LAZY max-propagation, the junction tree is max-consistent. From the max-consistent junction tree, a configuration of maximal probability cannot right away be determined using the approach described in Section 4.4, because clique and separator potentials are decomposed as sets of potentials. The approach described in Section 4.4 requires that the clique potentials have been computed. A configuration of all the variables of maximal probability is found by performing a *max configuration distribution* from the root of the junction tree:

Algorithm 4.6 (*Max configuration distribution*). Let C_i and C_j be adjacent cliques and let S be the separator between C_i and C_j. If *Max configuration distribution* is invoked on C_j from C_i, then:
(1) The configuration $\widehat{s}$ from C_i to C_j is absorbed.
(2) Determine a configuration of maximal probability of C_j (Algorithm 4.7).
(3) C_j invokes max configuration distribution on all adjacent cliques except C_i.

Absorption of the configuration $\widehat{s}$ proceeds by instantiating the variables of S in C_j to the configuration $\widehat{s}$. Algorithm 4.6 uses the *Find max configuration* algorithm:

Algorithm 4.7 (*Find max configuration*). Let C be a clique, let S be the separator between C and its parent clique, and let $\widehat{s}$ be the configuration of S passed to C. If *Find max configuration* is invoked on C, then:
(1) Set

$$\mathcal{R}_C = \Phi_C \cup \bigcup_{S' \in ne(C)} \Phi_{S'}.$$

(2) Instantiate all variables $\{X \in dom(\phi) \mid \phi \in \mathcal{R}_C, X \in S\}$ to the configuration $\widehat{s}$.
(3) Calculate

$$\widehat{c} = \arg \max_{\widehat{s}, X \in C \setminus \{S\}} \prod_{\phi \in \mathcal{R}_C} \phi.$$

(4) Return $\widehat{c}$.

The straightforward approach of just combining all the potentials associated with a clique in order to find the maximizing arguments of the clique potential is not necessarily the most efficient approach.

The max configuration $\widehat{r}$ of the variables in the root clique R of the junction tree can be determined by max eliminating all variables from the set of potentials associated with R and the subsets of potentials passed to R. If all intermediate potentials constructed

during the elimination is stored, then by processing the variables in the reverse order of the elimination, a max configuration can be obtained easily. The last eliminated variable X_1 is eliminated from the max marginal of X_1, the second to last eliminated variable X_2 is eliminated from a set of potentials Φ_{X_2} only including X_1 and X_2 in the domain, and so on. The configuration x_1 of X_1 in the max configuration can be determined from the max marginal of X_1, the configuration x_2 of X_2 can be obtained from Φ_{X_2} after X_1 is instantiated to x_1, and so on.

Let C be a clique adjacent to R and let S be the connecting separator. The sub-configuration of $\widehat{r}$ corresponding to the variables of S is distributed to C and the above procedure is repeated. This step is repeated for each adjacent clique of R.

The above improvement requires that all intermediate potentials are maintained, but it can still have a substantial lower space cost than representing the full joint of all clique variables. If space cost is a problem, then the set of intermediate potentials can be recomputed as needed instead of maintaining the entire set of intermediate potentials.

A configuration of maximal probability and the probability of this configuration can be determined by only performing a COLLECTEVIDENCE followed by a max configuration distribution. That is, the DISTRIBUTEEVIDENCE is unnecessary unless the max marginal distributions of all cliques are of interest.

4.6. Fast retraction of evidence

Cowell and Dawid [3] argue that a task of particular importance with respect to monitoring the quality of the probabilistic forecasts made by a system involves comparing evidence on a specific variable with its distribution given the evidence on all the other variables. Let ε be the set of evidence, let X be a variable instantiated by evidence to x, and let ε_X be the evidence function corresponding to $X = x$. Thus, the task of monitoring the quality of a system involves comparing the evidence $X = x$ with $P(X|\varepsilon \setminus \{X = x\})$. Efficient calculation of such distributions is supported by fast retraction of evidence.

The straightforward approach to fast retraction of a piece of evidence $X = x$ is to associate the evidence function ε_X with only one clique C. The distribution $P(X|\varepsilon \setminus \{X = x\})$ can be computed by marginalization of $C \setminus \{X\}$ from the combination of the potentials associated with C except ε_X. This will, however, eliminate the possibility to take full advantage of the independence relations induced by the evidence during message passing.

In the general case, the fast retraction of a piece of evidence $X = x$ inserted by instantiating all potentials including X in the domain does not reduce to removing the evidence function ε_X from a clique. A partial propagation can be necessary in the subtree of the junction tree containing $X = x$. To realize this, consider the following example.

Example 4.6. Let C_i and C_j be adjacent cliques separated by S, see Fig. 22. Let ε_{C_i} be the evidence passed to C_i from all its neighbours except S and let ε_{C_j} be the evidence passed to C_j from all its neighbours except S. Assume variable X is a member of cliques C_i and C_j only and assume—without loss of generality—that ε_X is the only evidence associated with C_i and C_j. Finally, assume that the evidence $\varepsilon = \varepsilon_{C_i} \cup \varepsilon_{C_j} \cup \varepsilon_X$ has been propagated

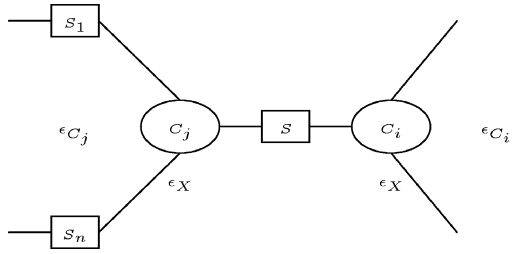

Fig. 22. Fast retraction of evidence on a variable X requires a partial propagation in the subtree of the junction tree containing X.

in the junction tree. From the set of potentials associated with clique C_j, the joint of C_j and ε can be calculated:

$$P(C_j, \varepsilon) = \varepsilon_X \prod_{\phi \in \Phi_{C_j}} \phi \prod_{S \in ne(C_j)} \prod_{\phi' \in \Phi_S} \phi_S.$$

Because independence relations induced by $X = x$ are exploited when calculating the message passed to C_i from C_j, fast retraction of the evidence on X from $P(C_j, \varepsilon)$ cannot always be done by just removing the evidence function ε_X. This implies that fast retraction of evidence on a variable might require a (partial) COLLECTEVIDENCE to the clique from which the evidence is fast retracted.

If fast retraction of evidence is based on LAZY cautious entering of evidence, then fast retraction of evidence becomes simple. Let X be a variable instantiated by evidence to x. The independence relations induced by $X = x$ are not exploited during message passing between the cliques containing X in the domain. Thus, fast retraction of the evidence on X just amounts to removing ε_X. Notice that LAZY cautious entering of evidence makes it possible to fast retract a piece of evidence from more than one clique.

4.7. Incremental evidence

Incrementality with respect to evidence enables an inference architecture to update its representation when new evidence arrives without recomputing everything from the initial representation [4]. Evidence incrementality is possessed by most inference architectures. In this section we will describe how LAZY propagation can take advantage of incremental evidence.

4.7.1. The traditional approach

When evidence on a set of variables has to be propagated in a consistent junction tree, it is not necessary to recompute every message in the junction tree.

One of the most efficient methods for handling incremental evidence in standard junction tree propagation architectures known to the authors is the method introduced by Dawid [7]. When new evidence ε arrives, a full round of message passing is performed in the subtree $\mathcal{T}^\varepsilon$ of the junction tree $\mathcal{T}$ containing the evidence. Next, ε is distributed to the remaining

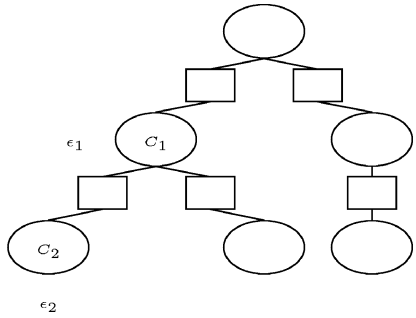

Fig. 23. Messages not invalidated by evidence do not have to be recalculated.

parts of $\mathcal{T}$ by distributing from $\mathcal{T}^{\varepsilon}$ to all the cliques of $\mathcal{T}$ not in $\mathcal{T}^{\varepsilon}$. A message which has to be recomputed due to the insertion of evidence is referred to as an *invalid message*.

Example 4.7. Consider the junction tree depicted in Fig. 23 and assume $\varepsilon = \{\varepsilon_1, \varepsilon_2\}$. In this case $\mathcal{T}^{\varepsilon}$ consists of cliques C_1 and C_2. First, two messages will be passed between C_1 and C_2. Next, messages are distributed from C_1 to the remaining cliques in $\mathcal{T}$. Six messages have to be passed between cliques in order to make $\mathcal{T}$ consistent.

The main weakness of the approach described above is that the entire content of an invalid message has to be recalculated even though the evidence only influences a small or no part of the message. Furthermore, a message passed over a separator S is recalculated even if the evidence is on a variable in the domain of S.

Recently, two other approaches to handling incremental evidence efficiently have been proposed. Lin and Druzdzel [20] introduce an updating method based on relevance-based reasoning and Darwiche [6] introduces dynamic jointrees for reconfigurating the computational structure dynamically as the query changes.

4.7.2. LAZY *propagation of incremental evidence*

In Section 3.4 we described how the independence relations induced by evidence obtained before the initial propagation are exploited to reduce the cost of inference. The independence relations induced by incremental evidence can also be exploited to reduce the cost of inference.

The following theorem describes an important property of insertion of evidence in the LAZY propagation architecture.

Theorem 4.8. *Let $N = (G, \mathcal{P})$ be a Bayesian network and let $\mathcal{T}$ be a junction tree representation of N made consistent with* LAZY *propagation. If ϕ is a potential associated with $\mathcal{T}$, then evidence on a variable $X \in dom(\phi)$ cannot imply that*:

$$\varepsilon_X \cdot \phi = \varepsilon_X \cdot \prod_{i=1}^{n} \phi_i,$$

where $\phi_1, \ldots, \phi_n$ are potentials or combinations of potentials of $\mathcal{P}$ and $n > 1$.

Proof. Let ϕ be a potential and let $X \in dom(\phi)$. Assume evidence on X has the effect that ϕ can be split up into a set of sub-potentials $\phi_1, \ldots, \phi_n$ where ϕ_i for all i is either a potential of $\mathcal{P}$ or produced as the result of eliminating a set of variables from a combination of potentials of $\mathcal{P}$. Potentials are only combined when variables are eliminated and therefore the only variables which can split ϕ up into a set of potentials are variables eliminated from the combination of $\phi_1, \ldots, \phi_n$. This conflicts with the fact that X is in the domain of ϕ. □

Let $\mathcal{T}$ be a consistent junction tree and assume—without loss of generality—that $\varepsilon = \{X = x\}$ has not yet been inserted into $\mathcal{T}$. The potentials Φ_X including X in the domain are situated in a connected subtree $\mathcal{T}^\varepsilon$ of the junction tree. The domain of each potential $\phi \in \Phi_X$ is decreased to reflect the evidence. When the domains of all potentials $\phi \in \Phi_X$ have been decreased, $\mathcal{T}^\varepsilon$ is consistent. In order to make the entire junction tree consistent a DISTRIBUTEEVIDENCE from $\mathcal{T}^\varepsilon$ to the remaining parts of the junction tree is necessary. Instead of recomputing all messages passed in $\mathcal{T} \setminus \mathcal{T}^\varepsilon$ during the previous propagation of evidence only the potentials invalidated by ε should be recomputed.

Let S be a separator connecting two adjacent cliques C_i and C_j in a consistent junction tree and let $S_1, \ldots, S_n$ be the remaining neighbours of C_j, see Fig. 24. Let Φ_S be the set of potentials passed from C_j to C_i over S during the propagation performed previous to the insertion of ε. Each potential $\phi \in \Phi_S$ is the result of eliminating a set of variables $V_\phi \subseteq C_j \setminus S$. Let $\mathcal{R}_S$ be the set of potentials relevant for the computation of Φ_S and let $\mathcal{R}_\phi \subseteq \mathcal{R}_S$ be the set of potentials combined in the process of eliminating the variables of V_ϕ. For each potential $\phi \in \Phi_S$ a computation tree can be constructed. The root of the computation tree is ϕ and each leaf node of the tree corresponds to a potential of $\mathcal{R}_\phi$. Internal nodes corresponds to either the combination of potentials or elimination of a variable. Let $\mathcal{R}^*_S$ be the set of potentials associated with C_j and neighbours except S after the insertion of ε. For each potential $\phi \in \Phi_S$ we determine whether or not the leaves of the computation tree corresponding to the elimination of variables V_ϕ from $\mathcal{R}^*_S$ are the same as the leaves of the computation tree corresponding to the elimination of variables V_ϕ from $\mathcal{R}_S$. If the leaves are not the same, then ϕ is assumed to be invalidated by evidence. Thus, ϕ

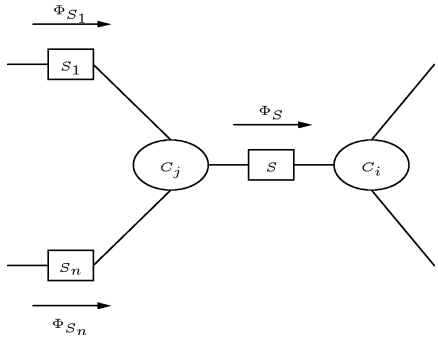

Fig. 24. Not all potentials in a message have to be recalculated when a message Φ_S is invalidated by a piece of evidence.

is not included in the new message Φ_S^*. The elimination of variables V_ϕ has to be repeated producing a new set of potentials.

There are two circumstances under which a potential $\phi \in \Phi_S$ can be invalidated by ε. Either X was eliminated in the process of constructing ϕ or ε has induced new dependence relations between variables eliminated in the process of constructing ϕ and other variables.

The adjusted absorption algorithm is:

Algorithm 4.8 (*Incremental absorption*). Let C_i and C_j be adjacent cliques, let S be the separator between C_i and C_j, and let ε be the new evidence. If *Incremental absorption* is invoked on C_j from C_i, then:

(1) Set

$$\mathcal{R}_S^* = \Phi_{C_j} \cup \bigcup_{S' \in ne(C_j)\setminus\{S\}} \Phi_{S'}.$$

(2) Let Φ_S be the set of potentials passed from C_j to C_i over S during the previous propagation.
(3) Let $\mathcal{R}_S$ be the set of potentials relevant for the computation of Φ_S during the previous propagation.
(4) **For** each $\phi \in \Phi_S$
   (a) Let $V_\phi \subseteq C_j \setminus S$ be the set of variables eliminated in the process of producing ϕ.
   (b) Let $\mathcal{R}_\phi \subseteq \mathcal{R}_S$ be the set of potentials combined in the process of eliminating V_ϕ.
   (c) Let $\mathcal{R}_\phi^* \subseteq \mathcal{R}_S^*$ be the new set of potentials relevant for eliminating V_ϕ.
   (d) **If** $\mathcal{R}_\phi \neq \mathcal{R}_\phi^*$, **then**
      (i) Mark ϕ as invalid.
      else
      (i) Mark ϕ as valid.
(5) Recompute all invalidated potentials by marginalization of the corresponding variables.
(6) Let Φ_S^* be the set of potentials obtained including the valid potentials.
(7) Associate Φ_S^* with S as the set of potentials passed from C_j to C_i.

Notice that the set of potentials in the new message is the same regardless of the order in which the variables are eliminated. The complexity of the computations producing the message can, however, vary considerably with the elimination order.

In the LAZY propagation architecture the decomposition of clique and separator potentials makes it possible to increase the computational efficiency of propagating incremental evidence. If a message passed between two cliques is invalidated by new evidence, then it is not always necessary to recompute the entire message. Only the potentials in the old message which are invalidated by the new evidence have to be recomputed.

The description made above is for the case of a single piece of evidence and where the passing clique C_j only has one neighbour in the subtree $\mathcal{T}^\varepsilon$. The generalization of

the approach to the case of multiple pieces of evidence and multiple adjacent cliques is straightforward.

Notice that LAZY propagation is able to avoid recalculating messages in the DISTRIBUTEEVIDENCE phase of propagation without comparing the contents of the potentials. Thus, incremental evidence can be handled more efficiently in the LAZY propagation architecture than in any of the standard junction tree propagation architectures.

4.8. Independence of causal influence

In general, the efficiency of algorithms for probabilistic inference in Bayesian networks can be improved by exploiting independence of causal influence. This is also true for LAZY propagation. A large number of different approaches to exploiting independence of causal influence to reduce the cost of inference in Bayesian networks exists. Examples are parent divorcing [22], temporal transformation [10,11], the local expression language [5], heterogeneous factorization [32,33], ci-elimbel [23], the factorized representation [29], and LAZY parent divorcing [21].

Different approaches to exploiting independence of causal influence with a junction tree inference algorithm can be considered. One approach is to use a method like parent divorcing or temporal transformation to change the structure of the Bayesian network before the junction tree is constructed. Another approach is to create a junction tree from the Bayesian network and then change the set of potentials associated with the cliques of the junction tree. The first approach does not require adjustments to the inference algorithm while the second approach might require adjusting the inference algorithm to cope with the changes in the sets of potentials associated with the cliques of the junction tree.

It is a simple task to extend LAZY propagation to take advantage of any of the methods for exploiting independence of causal influence mentioned above. Changing the structure of the Bayesian network has no impact on the inference algorithm and changing the sets of potentials associated with the cliques of the junction tree only has a limited impact on the inference algorithm. Instead of associating a conditional probability distribution $P(E|C_1,\ldots,C_n)$ of an effect variable E given its parent cause variables C_i (for $i = 1,\ldots,n$) with the appropriate clique C of the junction tree, the factorized representation, LAZY parent divorcing, and heterogeneous factorization, for example, introduce a set of potentials $\Phi = \{\phi_1,\ldots,\phi_n\}$ reflecting a decomposition of the distribution.

The factorized representation, LAZY parent divorcing, and heterogeneous factorization offer a decomposition of conditional probability distributions and LAZY propagation exploits a decomposition of clique and separator potentials. From this it seems straightforward that LAZY propagation can readily be extended to take advantage of the more fine grained decomposition offered by factorized representation, LAZY parent divorcing, and heterogeneous factorization, respectively. The factorized representation and LAZY parent divorcing do not impose any constraint on the order in which variables can be eliminated while heterogeneous factorization do impose constraints on the elimination order.

Madsen and D'Ambrosio [21] report on empirical results obtained with the LAZY propagation architecture extended with LAZY parent divorcing and factorized representation to

take advantage of independence of causal influence to reduce the cost of inference. The results indicate that substantial savings can be expected.

5. Discussion

The results of the empirical evaluations related to time efficiency of LAZY propagation indicate that although some evidence may increase time costs, the overall effect of instantiating variables is a decrease of time costs. With many variables instantiated, LAZY propagation clearly outperforms standard propagation architectures. Furthermore, the results of the empirical evaluations related to space efficiency indicate that the space efficiency of LAZY propagation is considerably better than the space efficiency of standard propagation architectures.

The results of the empirical evaluation of LAZY propagation indicate that the LAZY propagation architecture offers the most substantial improvements relative to the standard architectures on junction trees with large clique sizes. Junction trees with large clique sizes are also the most important to be able to manage as most real-world Bayesian networks tend to have corresponding junction trees with largest cliques of considerable size and performance improvements of inference in small junction trees is not particular interesting.

For ease of exposition the LAZY propagation architecture has been described in the context of junction trees, but notice that the principles of lazy evaluation can be applied to other computational tree structures. One topic of current research is to consider other computational structures as the basis of LAZY evaluation. Any computational tree structure maintaining the (in)dependence relations of the Bayesian network can be used as the underlying structure for controlling the computations performed by the LAZY propagation algorithm.

The LAZY propagation architecture offers a tradeoff between the time used on on-line and off-line triangulation. During propagation of evidence each clique passes messages to adjacent cliques. Each message is calculated by elimination of variables. The internal elimination order of each clique is determined on-line in order to exploit barren variables and independence relations induced by evidence to improve the efficiency of inference. The off-line produced triangulation serves the purpose of creating a partial order on the set of possible on-line elimination orderings. The off-line produced triangulation is transformed into a junction tree. The structure of the junction tree enforces a partial order on the set of possible on-line elimination orderings. The size of the largest clique is an upper bound on the worst-case complexity of performing LAZY propagation in the junction tree.

A motivation for considering different underlying computational structures of LAZY propagation is to increase the number of degrees of freedom when performing the on-line triangulation locally in each clique. Additional degrees of freedom during on-line triangulation might decrease the number of fill-in edges added. The largest number of degrees of freedom is obtained if the entire Bayesian network is contained in a single clique. It is, however, generally agreed that maintaining a secondary computational

structure such as a junction tree is superior to direct computation when computing all marginals or if numerous sets of evidence is considered. Changing the underlying computational structure of LAZY propagation is as mentioned a topic of current research.

6. Conclusion

In this paper we have presented an algorithm for probabilistic inference in Bayesian networks based on lazy evaluation. The proposed architecture improves both the conceptual understanding of and the performance of inference in Bayesian networks.

A number of empirical evaluations involving large real-world Bayesian networks have been performed to emphasize the efficiency improvements offered by the proposed architecture. The empirical evaluations which are based on non-optimized implementations of the HUGIN, Shafer–Shenoy, and LAZY propagation architectures are designed to determine the time and space efficiency of LAZY propagation relatively to HUGIN and Shafer–Shenoy propagation. Even though the implementations of the Shafer–Shenoy and HUGIN architectures used for the comparison are simple, the results of the empirical evaluation are convincing.

The LAZY propagation architecture enlarges the class of tractable Bayesian networks as both the time and space costs of this architecture are smaller than the costs of the HUGIN and Shafer–Shenoy architectures.

Acknowledgement

This research was performed while Anders L. Madsen was a Ph.D.-student at Department of Computer Science, Aalborg University, Denmark. This research was partly supported by the Danish Natural Science Research Council grant 9601649.

Appendix A. Size of initialized junction tree

Figs. A.1–A.3 show plots of the average size of the initialized junction tree in the HUGIN, Shafer–Shenoy, and LAZY propagation architectures as a function of the size of the evidence set.

Appendix B. Size of largest potential

Figs. B.1–B.3 show plots of the average size of the largest potential calculated during inference in the HUGIN, Shafer–Shenoy, and LAZY propagation architectures as a function of the size of the evidence set.

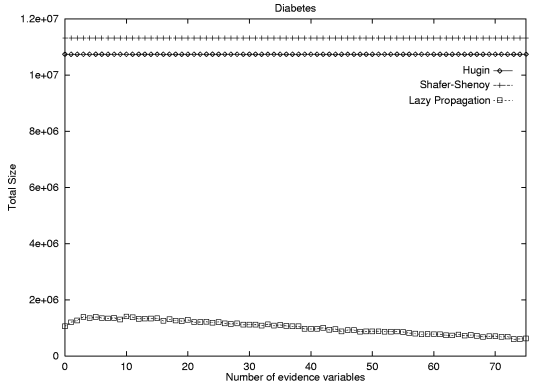

Fig. A.1. A plot of the average size of the initialized junction tree for the Diabetes network as a function of the number of variables instantiated.

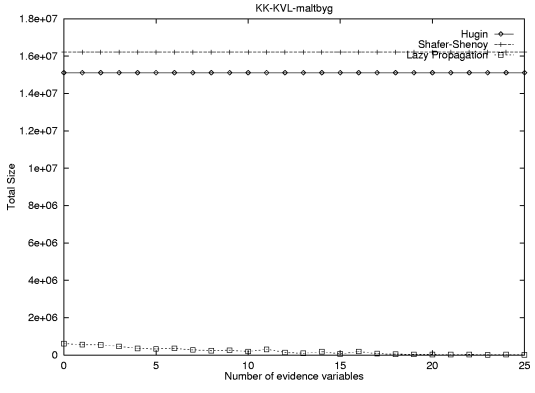

Fig. A.2. A plot of the average size of the initialized junction tree for the KK network as a function of the number of variables instantiated.

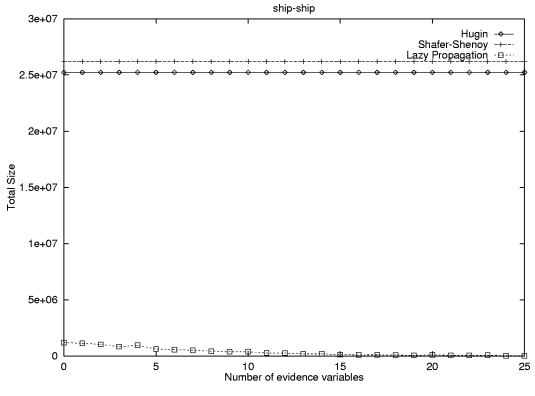

Fig. A.3. A plot of the average size of the initialized junction tree for the ship-ship network as a function of the number of variables instantiated.

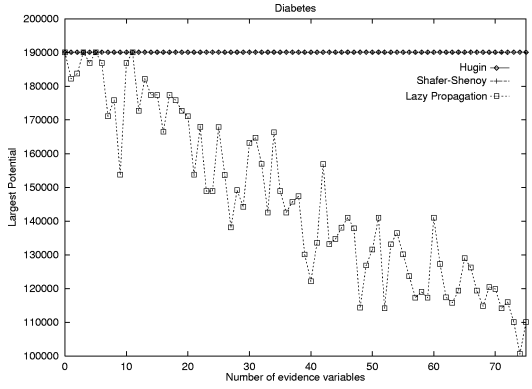

Fig. B.1. A plot of the average size of the largest potential computed during propagation of evidence in the Diabetes network as a function of the number of variables instantiated.

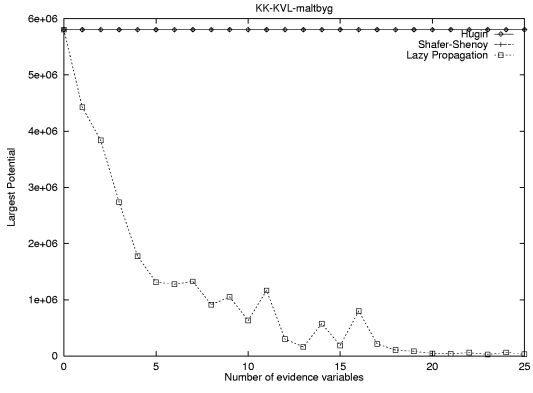

Fig. B.2. A plot of the average size of the largest potential computed during propagation of evidence in the KK network as a function of the number of variables instantiated.

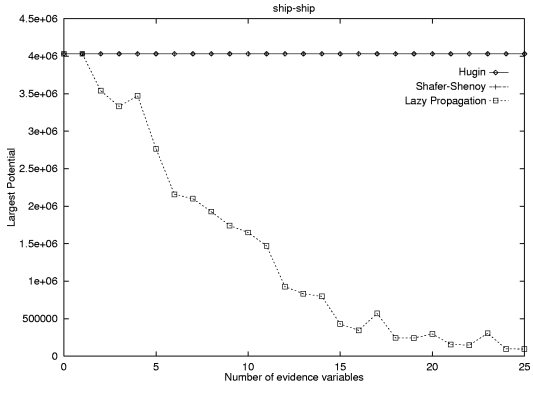

Fig. B.3. A plot of the average size of the largest potential computed during propagation of evidence in the ship-ship network as a function of the number of variables instantiated.

References

[1] C. Cannings, E.A. Thompson, H.H. Skolnick, Probability functions on complex pedigrees, Adv. Appl. Probab. 10 (1978) 26–61.
[2] G.F. Cooper, The computational complexity of probabilistic inference using Bayesian belief networks, Artificial Intelligence 42 (2–3) (1990) 393–405.
[3] R.G. Cowell, A.P. Dawid, Fast retraction of evidence in a probabilistic expert system, Statist. Comput. 2 (1992) 37–40.
[4] B. D'Ambrosio, Incremental probabilistic inference, in: Proc. 9th Conference on Uncertainty in Artificial Intelligence, Washington, DC, 1993, pp. 301–308.
[5] B. D'Ambrosio, Local expression language for probabilistic dependence, Internat. J. Approx. Reason. 13 (1) (1995) 61–81.
[6] A. Darwiche, Dynamic jointrees, in: Proc. 14th Conference on Uncertainty in Artificial Intelligence, Madison, WI, 1998, pp. 97–104.
[7] A.P. Dawid, Applications of a general propagation algorithm for probabilistic expert systems, Statist. Comput. 2 (1992) 25–36.
[8] R. Dechter, Bucket elimination: A unifying framework for probabilistic inference, in: Proc. 12th Conference on Uncertainty in Artificial Intelligence, Portland, OR, 1996, pp. 211–219.
[9] D. Geiger, T.S. Verma, J. Pearl, d-separation: From theorems to algorithms, in: M. Henrion, R.D. Shachter, L.N. Kanal, J.F. Lemmer (Eds.), Uncertainty in Artificial Intelligence 5, Elsevier Science, Amsterdam, 1990, pp. 139–148.
[10] D. Heckerman, Causal independence for knowledge acquisition and inference, in: Proc. 9th Conference on Uncertainty in Artificial Intelligence, Washington, DC, 1993, pp. 122–127.
[11] D. Heckerman, J.S. Breese, A new look at causal independence, in: Proc. 10th Conference on Uncertainty in Artificial Intelligence, Seattle, WA, 1994, pp. 286–292.
[12] F. Jensen, S.K. Andersen, Approximations in Bayesian belief universes for knowledge based systems, in: Proc. 6th Workshop on Uncertainty in Artificial Intelligence, Cambridge, MA, 1990.
[13] F.V. Jensen, Cautious propagation in Bayesian networks, in: Proc. 11th Conference on Uncertainty in Artificial Intelligence, Montreal, Quebec, 1995, pp. 323–328.
[14] F.V. Jensen, An Introduction to Bayesian Networks, UCL Press, London, 1996.
[15] F.V. Jensen, S.L. Lauritzen, K.G. Olesen, Bayesian updating in causal probabilistic networks by local computations, Computational Statistics Quarterly 4 (1990) 269–282.
[16] F.V. Jensen, K.G. Olesen, S.K. Andersen, An algebra of Bayesian belief universes for knowledge-based systems, Networks 20 (1990) 637–659.
[17] U. Kjærulff, Inference in Bayesian networks using nested junction trees, Learning in Graphical Models 89 (1998) 51–74.
[18] S.L. Lauritzen, D.J. Spiegelhalter, Local computations with probabilities on graphical structures and their application to expert systems, J. Roy. Statist. Soc. B. 50 (2) (1988) 157–224.
[19] Z. Li, B. D'Ambrosio, Efficient inference in Bayes networks as a combinatorial optimization problem, Internat. J. Approx. Reason 11 (1) (1994) 55–81.
[20] Y. Lin, M.J. Druzdzel, Relevant-based sequential evidence processing in Bayesian networks, in: Proc. 11th International FLAIRS Conference, 1998, pp. 446–450.
[21] A.L. Madsen, B. D'Ambrosio, Independence of causal influence and lazy propagation, in: Proc. 5th European Conference on Symbolic and Quantitative Approaches to Reasoning with Uncertainty, London, UK, 1999, pp. 293–304.
[22] K.G. Olesen, U. Kjærulff, F. Jensen, F.V. Jensen, B. Falck, S. Andreassen, S.K. Andersen, A MUNIN network for the median nerve—A case study on loops, Appl. Artificial Intelligence 3 (1989) 255–277. Special issue: Towards causal AI models in practice.
[23] I. Rish, R. Dechter, On the impact of causal independence, in: Stanford Spring Symposium on Interactive and Mixed-Initiative Decision-Theoretic Systems, 1998, pp. 101–108.
[24] T. Schmidt, P.P. Shenoy, Some improvements to the Shenoy–Shafer and Hugin architectures for computing marginals, Artificial Intelligence 102 (2) (1998) 323–333.
[25] R. Shachter, Evaluating influence diagrams, Oper. Res. 34 (6) (1986) 871–882.
[26] R.D. Shachter, An ordered examination of influence diagrams, Networks 20 (5) (1990) 535–563.

[27] G.R. Shafer, P.P. Shenoy, Probability propagation, Ann. Math. Artificial Intelligence 2 (1990) 327–352.
[28] P.P. Shenoy, Binary join trees for computing marginals in the Shenoy–Shafer architecture, Internat. J. Approx. Reason. 17 (2–3) (1997) 239–263.
[29] M. Takikawa, B. D'Ambrosio, Multiplicative factorization of noisy-max, in: Proc. 15th Conference on Uncertainty in Artificial Intelligence, Stockholm, Sweden, 1999, pp. 622–630.
[30] H. Xu, Computing marginals from the marginal representation in Markov trees, in: Proc. International Conference on Information Processing and Management of Uncertainty in Knowledgebased Systems (IPMU), 1994, pp. 275–280.
[31] N.L. Zhang, D. Poole, Intercausal independence and heterogeneous factorization, in: Proc. 10th Conference on Uncertainty in Artificial Intelligence, Seattle, WA, 1994, pp. 606–614.
[32] N.L. Zhang, D. Poole, Exploiting causal independence in Bayesian network inference, J. Artificial Intelligence Res. 5 (1996) 301–328.
[33] N.L. Zhang, L. Yan, Independence of causal influence and clique tree propagation, in: Proc. 13th Conference on Uncertainty in Artificial Intelligence, Providence, RI, 1997, pp. 472–280.